

UR HTTP Server Protocol Stack

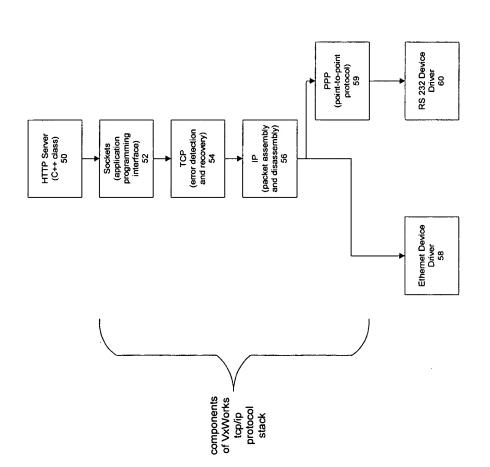


FIG. 4

UR HTTP Server Data Flow

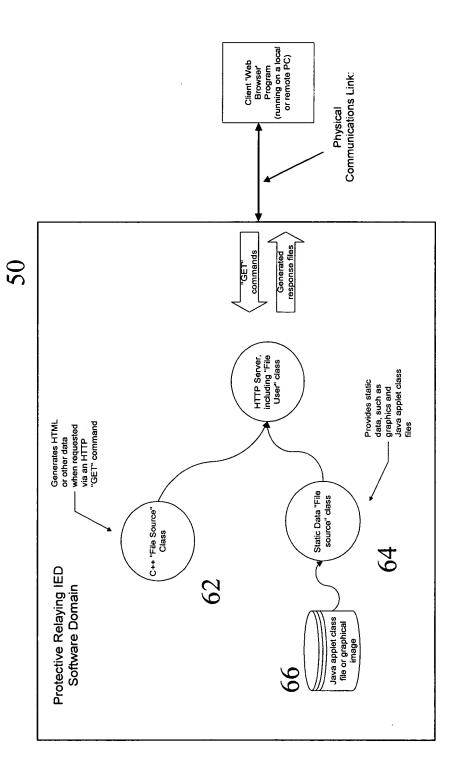


FIG.



FIG. 6 Source Code

Listing 1: COM_Webserver.h	1
Listing 2: COM_Webserver.cpp	2
Listing 3: UTL FileSource.h	
Listing 4: UTL FileSource.cpp	
Listing 5: UTL WebPage.h	
Listing 6: UTL WebPage.cpp	
Listing 7: UTL FileUser.h	
Listing 8: UTL_FileUser.cpp	
Listing 6. OTL_1 neoser.epp	

Listing 1: COM_Webserver.h

```
* Copyright (C) General Electric Co. GE Confidential and Proprietary
 \star DESCRIPTION This file contains the Modbus/TCP communications port sub-class.
#ifndef COM_WebServer_H
#define COM WebServer_H
// INCLUDES
#include "COM_Hardware.h"
#include "SYS_DPRAM.h"
#include <assert.h>
#include "DB_NotificationSource.h"
                                         // maximum number of simultaneous connections
#define MAX HTTP_CONNECTIONS
                                   3
class UTL_lmsTimer;
// Web server task -- handles HTTP protocol connections over TCP/IP, so you
// web site. Other UTL_WebPage objects may be accessed through the menu structure
// or by specifying the URL for each page. The UTL WebPage objects are distributed
// through the UR firmware files, so each one is near the data it requires.
// You can also read non-webpage UTL_FileSource files with the web browser, but
// they don't appear in menus.
// <BR>
// For more information about web pages and other files, see UTL_WebPage and
// UTL_FileUser.
class COM_WebServer : public DB_NotificationSource
public:
                                                       // Constructor
    COM_WebServer();
                                                           // Destructor
    virtual ~COM WebServer();
    virtual void sendFrame(unsigned char *buffer, UR_UINT16 length, int con_sFd );
    void connect_Task(void);
static int call_connect_Task(COM_WebServer *);
    static int call_read_Task( COM_WebServer * obj, int connectionNumber );
void read_Task( int connectionNumber );
    void acceptNotification(DB_NotificationSource *source, int param);
    // Get the number of active connections
    int getConnectionCount(void) { return connectionCount; }
    // Returns a pointer to the only object of this class. static COM_WebServer * const find( void )
        assert( the COM_WebServer );
        return the COM_WebServer;
    }
```

```
// Deletes all instances of this class
    static void deleteAll(void)
        if ( the COM WebServer )
             delete the COM WebServer;
protected:
                                               // number of currently active connections
    int connectionCount;
private:
    // Pointer to what should be the only object of this class
    static COM_WebServer * the_COM_WebServer;
    char tName[64];
                                               // task name string
                                               // socket file descriptor
    int sFd;
    unsigned char *transmitBufferPtr;
                                               // buffer pointer
                                               // indicates when initialization is complete
    UR BOOLEAN isInitialized;
    int b64 decode( const char* str, unsigned char* space, int size );
    static const int b64_decode_table[256]; // table used for base-64 decoding
    char clientPassword[100];
                                               // storage for decoded password
    void notFoundPage(int connected_sFd); // display "page not found" page
UTL_1msTimer * connectionTimers[MAX_HTTP_CONNECTIONS]; // timers to kill dead
connections
                                                   // socket descriptors -- 1 per connection
    int connected sFd[MAX_HTTP_CONNECTIONS];
                                               // number of tasks which are running (helps
    int numRunningTasks;
with shutdown)
                                               // true means we're shutting down
    bool pleaseKillMe;
}:
#endif
Listing 2: COM_Webserver.cpp
                           *************
 * Copyright (C) General Electric Co. GE Confidential and Proprietary
 * DESCRIPTION Web server class and related stuff
// DEFINES
                             80 // registered port number for http
#define SERVER_PORT_NUM
                             20 // port inactivity timeout value in seconds
#define SOCKET TIMEOUT
                                  // non-zero means show debug messages
#define DEBUG HTTP
// INCLUDES
#include "COM_WebServer.h"
#include "COM_ModbusApplication.h"
#include "DB_IPAddress.h"
#include "DB_Text.h"
#include "SYS_Data.h"
#include "UTL_MathFunctions.h"
#include "UTL_CardManager.h"
#include "SYS_Application.h"
#include "UTL_lmsTimer.h"
#include "UTL_TaskDataBlock.h"
#include "UTL WatchDog.h"
#include <taskLib.h>
#include <sysLib.h>
#include <vxWorks.h>
#include <stdioLib.h>
#include <vxwSemLib.h>
#include <ioLib.h>
#include <sockLib.h>
#include <inetLib.h>
#include <strLib.h>
#include <netinet\tcp.h>
#include <string.h>
#include <assert.h>
#include "UTL FileUser.h"
#include "UTL_FileSource.h"
#include "UTL_StaticFile.h"
#include "UTL_WebPage.h"
```

```
*******************
const char GifBug[] = {
 "\x66\x00\x00\x99\x00\xCC\x00\x00\xFF\x33\x00\x33\x00\x33\x33\x00\x66\x33\x00\x99"
"\x33\x00\xCC\x33\x00\xFF\x66\x00\x06\x00\x33\x66\x00\x66\x66\x00\x99\x66\x00\xCC\x66"
"\x00\xFF\x99\x00\x00\x99\x00\x33\x99\x00\x66\x99\x00\x99\x99\x00\xCC\x99\x00\xFF\xCC\x00"
"\x00\xCC\x00\x33\xCC\x00\x66\xCC\x00\x99\xCC\x00\xCC\x00\xFF\xFF\x00\x00\xFF\x00\x33"
"\xFF\x00\x66\xFF\x00\x99\xFF\x00\xCC\xFF\x00\xFF\x00\x33\x33\x00\x33\x66\x00"
"\xCC\x33\x33\xFF\x66\x33\x00\x66\x33\x33\x66\x33\x66\x33\x99\x66\x33\xCC\x66\x33\xFF"
"\x99\x33\x00\x99\x33\x33\x99\x33\x66\x99\x33\x99\x33\xCC\x99\x33\xFF\xCC\x33\x00\xCC"
"\x33\x33\xCC\x33\x66\xCC\x33\x99\xCC\x33\xCC\xCC\x33\xFF\xFF\x33\x00\xFF\x33\x33\xFF\x33"
"\x66\xFF\x33\x99\xFF\x33\xCC\xFF\x33\xFF\x00\x66\x00\x66\x33\x00\x66\x66\x99"
"\x00\x66\xCC\x00\x66\xFF\x33\x66\x00\x33\x66\x66\x33\x66\x66\x33\x66\x99\x33\x66\xCC\x33"
"\x00\x99\x66\x33\x99\x66\x66\x99\x66\x99\x66\xCC\x99\x66\xFF\xCC\x66\x00\xCC\x66\x33"
"\xCC\x66\x66\xCC\x66\x99\xCC\x66\xCC\xC6\xFF\xFF\x66\x00\xFF\x66\x33\xFF\x66\x66\xFF"
"\x66\x99\xFF\x66\xCC\xFF\x66\xFF\x60\x99\x00\x99\x33\x00\x99\x66\x00\x99\x00\x99"
"\xCC\x00\x99\xFF\x33\x99\x00\x33\x99\x66\x33\x99\x66\x33\x99\x66\x33\x99\xFF"
"\x66\x99\x00\x66\x99\x33\x66\x99\x66\x99\x66\x99\x66\x99\xCC\x66\x99\xFF\x99\x00\x99"
"\x66\xCC\x99\x99\xCC\x99\xCC\x99\xFF\xFF\x99\x00\xFF\x99\x33\xFF\x99\x66\xFF\x99\x99"
"\xFF\x99\xCC\xFF\x99\xFF\x00\xCC\x00\xC0\x33\x00\xCC\x66\x00\xCC\x99\x00\xCC\x00" -
"\xCC\xFF\x33\xCC\x00\x33\xCC\x33\x33\xCC\x66\x33\xCC\x99\x33\xCC\xCC\x33\xCC\xFF\x66\xCC"
"\x00\x66\xCC\x33\x66\xCC\x66\x66\xCC\x99\x66\xCC\x66\xCC\x66\xCC\x66\xCC\x59\x66\xCC\x33"
"\x99\xCC\x66\x99\xCC\x99\x99\xCC\xCC\x99\xCC\xFF\xCC\xCC\x00\xCC\x33\xCC\xCC\x66\xCC"
"\xCC\x99\xCC\xCC\xCC\xCC\xCC\xFF\xFF\xCC\x00\xFF\xCC\x33\xFF\xCC\x66\xFF\xCC\x99\xFF\xCC"
"\xCC\xFF\xC0\xFF\x00\xFF\x00\x00\xFF\x33\x00\xFF\x66\x00\xFF\x99\x00\xFF\xCC\x00\xFF\xFF"
"\x33\xFF\x00\x33\xFF\x33\x33\xFF\x66\x33\xFF\x99\x33\xFF\xCC\x33\xFF\xFF\x66\xFF\x00\x66"
"\xFF\x33\x66\xFF\x66\x66\xFF\x99\x66\xFF\xCC\x66\xFF\xFF\x99\xFF\x33\x99\xFF"
"\x66\x99\xFF\x99\x99\xFF\xCC\x99\xFF\xCC\xFF\x00\xCC\xFF\x33\xCC\xFF\x66\xCC\xFF\x99"
"\xCC\xFF\xCC\xCC\xFF\xFF\xFF\xFF\xFF\x00\xFF\xFF\x33\xFF\xFF\x66\xFF\xFF\x99\xFF\xFF\xCC\xFF"
```

"\x45\x32\x2E\x30\x03\x01\xE8\x03\x00\x21\xF9\x04\x09\x14\x00\x83\x00\x2C\x00\x00\x00\x00" "\x20\x00\x20\x00\x07\x08\xDF\x00\x09\x1C\x04\x00\xC0\xC0\x82\x06\x0F\x16\x1C\xC8\xB0" "\xA1\x43\x82\x08\x21\x0A\x5C\xB8\xF0\xA1\x45\x8B\x09\x13\x4A\xBC\xC8\x11\x22\xC2\x88\x1E" "\x3B\x3E\x2C\x28\xED\x9A\x04\x09\xD7\xA4\x55\x14\x89\x51\x02\x80\x93\x6F\x5C\xBA\x64\xC9" "\x11\xE1\x49\x90\x34\x2F\xBE\x94\x10\x53\x63\xCE\x83\x0D\x01\xF4\x0C\x9A\x13\xA7\x47\x9F" "\x46\x75\x1A\xAC\x78\x92\x15\xAB\x93\x13\x97\xFA\x1C\xA9\xD1\xE6\xC9\xAB\x2E\xAB\x4E\x55" "\x98\x71\xE2\x55\xA7\x57\xBB\x86\x54\x9A\x10\x2B\xD6\xA8\x5B\xA9\x4A\xFC\xFA\x14\x2A\xC5" "\xB4\x0C\x57\x5A\x7D\x13\xF3\xA6\x56\x8C\x2B\x19\x36\x6D\x4B\x15\xAE\x45\xB3\x12\x7E\xEA" "\xDC\x7B\x53\xB0\xC3\x82\x80\xF3\x0A\xB6\xE9\x14\x6C\x52\x9A\x56\x13\x2B\x16\x19\xB9\x71" "\xDB\xAC\x7E\x0F\x2F\x04\x7C\xF6\xB1\x5A\x81\x7B\x2D\x43\x8D\x4A\x59\x23\xE7\xAB\x0A\x4B" "\x0F\x0C\xDD\x78\xF4\xC6\x9A\xA6\x4F\x07\x26\xDD\xD1\x27\x56\xB0\xA8\x81\xD6\x6E\x78\x9A" "\x68\xCD\x87\x80\x47\x1A\x1E\x2E\x30\x20\x00\x21\xF9\x04\x09\x14\x00\x83\x00\x2C\x00\x00" "\x00\x00\x20\x00\x20\x00\x07\x08\xE0\x00\x07\x09\x1C\x38\x10\x80\x41\x82\x06\x0F\x12\x5C" "\xC8\x30\x61\xC2\x41\x0A\x15\x42\x94\xC8\xB0\xE2\x42\x00\x02\x31\x36\xB4\xC8\x71\xA2\x43" "\x8A\x1D\x39\x02\x08\x14\xE8\x9A\x04\x09\xD7\x48\x6A\x0C\x79\x51\x23\x00\x09\x2F\x25\xBC" "\x81\x09\x93\x65\xCB\x88\x06\x4F\x3E\xCC\xB8\x52\xE4\xC1\x88\x27\x67\xAE\x74\xD8\x71\xA7" "\x51\xA1\x08\x31\x82\x2C\x48\x11\xA7\xC4\xA7\x3D\x8B\x62\x3C\xC9\x8A\xD5\x49\x9E\x49\x6D" "\x4E\x3C\xC9\x55\x67\x54\x9B\x2E\xB9\x56\xE5\xFA\x53\xAB\x47\x81\x5D\xBB\xF2\xFC\x5A\xF1" "\xE7\x54\xAA\x56\xBD\x2A\x65\x9B\x75\x6D\xD0\x99\x5E\x99\x5A\xDC\xB9\x10\xEE\xD5\x86\x4B" "\x3B\xA6\x95\x60\xB6\xA8\xDf\x9A\x85\x01\x0F\x0E\xAC\x35\xA1\x84\xAA\x63\xF9\x16\x76\x3C" "\x18\x26\x63\x9F\x31\x1F\x43\x26\x7B\x19\xF0\xDB\xCA\x84\x25\xFB\x1C\x08\x17\x72\x5C\xBD" "\x52\x09\x82\xFE\x7B\x76\x34\x69\xCD\xA6\x59\x4F\x74\xFD\x1A\xF4\x45\x91\x0C\xBB\x8E\xE5" "\x7A\xBB\xAD\x45\xDB\x1B\x0B\x0F\x4E\x4C\x9C\x65\x40\x00\x21\xF9\x04\x09\x14\x00\x83\x00" "\x2C\x00\x00\x00\x00\x20\x00\x20\x00\x07\x08\xE8\x00\x07\x09\x1C\x38\x10\x80\x41\x82\x06" "\x01\x10\x5C\xC8\x90\xE1\xC1\x83\x83\x1E\x2A\x6C\x48\xB1\xE0\xC4\x88\x02\x2F\x42\xAC\xB8" "\x30\x61\xC6\x84\x20\x3F\x5E\xE4\xD4\xD1\x83\xB4\x6B\x12\x24\x5C\x93\xE6\x61\x24\x49\x8B" "\x19\x25\x04\x4A\xF9\x46\xA6\x04\x97\x2F\x43\x46\x34\x98\xD2\xE3\xC7\x9C\x12\x63\x4A\xA8" "\xA9\xD1\x27\x45\x9F\x48\x89\x22\x54\xB8\xB1\x63\x53\x88\x3A\x77\xC2\xAC\x08\x55\x61\x4A" "\x56\xAC\x52\xFE\x9C\xEA\x10\x29\x80\x94\x60\x7B\x4E\x6C\xDA\xF5\xE7\xD7\xAB\x59\x7B\x8A" "\xC4\xB9\xF5\x67\xD8\xB0\x3B\xC7\xB2\x75\x99\x10\x2C\x56\xB0\x46\xA5\x02\x3D\xFB\xA6\xA6" "\xD8\x97\x24\xD1\x6A\x05\xFC\xF2\xAD\x04\xC2\x54\x79\x4A\xB8\x7B\x93\x2D\x49\x90\x8A\xC3" "\x42\x26\x3C\xF9\x2B\xD6\xBB\x90\x1D\x2F\x5D\x68\x78\x70\xDC\xC7\x23\xED\x5E\x06\x8B\x10" "\x34\xC1\xCE\xA4\xDB\x26\x1E\x88\xF6\x72\xDA\x82\x94\x27\xA2\x1E\x4C\x96\xA3\xEC\xC5\xAE" "\x5F\xD7\x7E\x3C\x68\xF6\xE1\xDD\xAB\xC3\x32\xA6\xAD\x99\x23\x6A\xC4\xC6\xDF\x22\x5F\x4E" "\x38\x20\x00\x21\xFE\xEF\x54\x68\x69\x73\x20\x47\x49\x46\x20\x66\x69\x6C\x65\x20\x77\x61" "\x73\x20\x61\x73\x73\x65\x6D\x62\x65\x65\x64\x20\x77\x69\x74\x68\x20\x47\x49\x46\x20\x43"

"\x6F\x6E\x73\x74\x72\x75\x63\x74\x69\x6F\x6E\x20\x53\x65\x74\x20\x66\x72\x6F\x6D\x3A\x0D" "\x0A\x0D\x0A\x41\x6C\x63\x68\x65\x6D\x79\x20\x4D\x69\x6E\x64\x77\x6F\x72\x6B\x73\x20\x49" "\x6E\x63\x2E\x0D\x0A\x50\x2E\x4F\x2E\x20\x42\x6F\x78\x20\x35\x30\x30\x0A\x42\x65\x65\" "\x74\x6F\x6E\x2C\x20\x4F\x6E\x74\x61\x72\x69\x6F\x0D\x0A\x4C\x30\x47\x20\x31\x41\x30\x0D" $"\x0A\x43\x41\x4E\x41\x44\x41\x2E\x0D\x0A\x0D\x0A\x54\x68\x69\x73\x20\x63\x6F\x6D\x6D\x65"$ "\x6E\x74\x20\x62\x6C\x6F\x63\x6B\x20\x77\x69\x6C\x6C\x20\x6E\x6F\x74\x20\x61\x70\x70\x65" "\x61\x72\x20\x69\x6E\x20\x66\x69\x6E\x65\x73\x20\x63\x72\x65\x61\x74\x65\x64\x20\x77\x69" "\x74\x68\x20\x61\x20\x72\x65\x67\x69\x73\x74\x65\x72\x65\x64\x20\x76\x65\x72\x73\x69\x6F" "\x6E\x20\x6F\x66\x20\x47\x49\x46\x20\x43\x6F\x6E\x73\x74\x72\x75\x63\x74\x69\x6F\x6E\x20" "\x53\x65\x74\x00\x21\xFF\x0B\x47\x49\x46\x43\x4F\x4E\x6E\x62\x31\x2E\x30\x02\x03\x00\x0E" "\x4F\x57\x53\x5C\x44\x65\x73\x6B\x74\x6F\x70\x5C\x41\x6E\x69\x6D\x5C\x69\x6E\x73\x74\x61" "\x6C\x6C\x5C\x62\x75\x67\x32\x5C\x4E\x65\x77\x2D\x31\x2E\x67\x69\x66\x00\x0E\x2F\x00\x02" "\x5C\x44\x65\x73\x6B\x74\x6F\x70\x5C\x41\x6E\x69\x6D\x5C\x69\x6E\x73\x74\x61\x6C\x6C\x5C" "\x73\x6B\x74\x6F\x70\x5C\x41\x6E\x69\x6D\x5C\x69\x6E\x73\x74\x61\x6C\x6C\x5C\x62\x75\x67" "\x32\x5C\x4E\x65\x77\x2D\x33\x2E\x67\x69\x66\x00\x00\x3B" const char UR GridClass[] = "\xCA\xFE\xBA\xBE\x00\x03\x00\x2D\x02\x54\x08\x01\x51\x08\x01\x53\x08\x01" "\x54\x08\x01\x85\x08\x01\x86\x08\x01\x88\x08\x01\x89\x08\x01\x8A\x08\x01\x8E\x08\x01\x8F" "\x08\x01\x90\x08\x01\x91\x08\x01\x92\x08\x01\x94\x08\x01\x95\x08\x01\x96\x08\x01\x98\x08" "\x01\xA8\x01\\xA9\x08\x01\xAA\x08\x01\xAB\x08\x01\xAC\x08\x01\xAE\x08\x01\xB1\x08\x01" "\xB2\x08\x01\xB9\x08\x01\xBC\x08\x01\xC3\x08\x01\xC8\x08\x01\xCF\x08\x01\xD2\x08\x01\xD8" "\x08\x01\xD9\x08\x01\xDA\x08\x02\x21\x08\x02\x22\x08\x02\x35\x08\x02\x53\x07\x01\xAF\x07" "\x01\xB4\x07\x02\x01\x07\x02\x02\x07\x02\x03\x07\x02\x04\x07\x02\x05\x07\x02\x06\x07\x02" "\x07\x07\x02\x08\x07\x02\x09\x07\x02\x0A\x07\x02\x0B\x07\x02\x0E\x07\x02\x0D\x07\x02\x0E" "\x07\x02\x0F\x07\x02\x10\x07\x02\x11\x07\x02\x12\x07\x02\x13\x07\x02\x14\x07\x02\x15\x07" "\x02\x16\x07\x02\x17\x07\x02\x18\x07\x02\x19\x07\x02\x1A\x07\x02\x1B\x07\x02\x1C\x07\x02" $"\x1D\x07\x02\x1E\x07\x02\x1F\x07\x02\x20\x0A\x00\x2A\x00\xD4\x0A\x00\x2D\x00\xD4\x0A\x00"$ "\x09\x00\x28\x00\xE0\x0A\x00\x31\x00\xE1\x0A\x00\x31\x00\xE2\x0A\x00\x2E\x0A\x00\xE3\x0A\x00" "\x2E\x00\xE4\x0A\x00\x44\x00\xE5\x0A\x00\x44\x00\xE6\x09\x00\x28\x00\xE7\x09\x00\x28\x00" "\xE8\x09\x00\x28\x00\xE9\x09\x00\x28\x00\xEA\x09\x00\x2F\x00\xEB\x0A\x00\x2C\x0A"

"\x00\x3C\x00\xED\x09\x00\x28\x00\xEE\x09\x00\x28\x00\xEF\x0A\x00\x30\x00\xF0\x0A\x00\x40" "\x00\xF1\x09\x00\x28\x00\xF2\x09\x00\x28\x00\xF3\x0A\x00\x24\x00\xF4\x0A\x00\x31\x00\xF5" "\x0A\x00\x2F\x00\xF6\x0A\x00\x32\x00\xF6\x0A\x00\x33\x00\xF6\x0A\x00\x43\x00\xF6\x09\x00" "\x28\x00\xF7\x09\x00\x28\x00\xF8\x09\x00\x28\x00\xF9\x09\x00\x28\x00\xFA\x09\x00\x28\x00" "\xFB\x09\x00\x28\x00\xFC\x09\x00\x28\x00\xFD\x09\x00\x35\x00\xFE\x09\x00\x28\x00\xFF\x09" $"\times00\times28\times01\times00\times00\times00\times39\times01\times01\times00\times30\times01\times00\times28\times001\times00\times28\times01\times00\times28\times01\times00\times28$ $"\x01\x04\x0A\x00\x30\x01\x05\x0A\x00\x30\x01\x06\x0A\x00\x37\x01\x07\x0A\x00\x48\x01\x08$ " $"\x0A\x00\x28\x01\x0A\x00\x2C\x01\x0A\x0A\x00\x33\x01\x0B\x0A\x00\x2A\x01\x0C\x0A\x00"$ "\x2D\x01\x0D\x0A\x00\x2E\x01\x0E\x0A\x00\x33\x01\x0F\x0A\x00\x30\x01\x10\x0A\x00\x49\x01" "\x00\x35\x01\x16\x09\x00\x35\x01\x17\x09\x00\x28\x01\x18\x09\x00\x32\x01\x19\x09\x00\x28" $"\x0A\x00\x28\x01\x1F\x0A\x00\x40\x01\x20\x0A\x00\x46\x01\x21\x0A\x00\x31\x01\x22\x09\x00"$ "\x28\x01\x23\x09\x00\x28\x01\x24\x0A\x00\x43\x01\x25\x09\x00\x28\x01\x26\x09\x00\x28\x01" "\x27\x09\x00\x28\x01\x28\x09\x00\x28\x01\x29\x09\x00\x28\x01\x2A\x0A\x01\x2B\x09" $"\times00\times45\times01\times2C\times0A\times00\times40\times01\times2D\times0A\times00\times3E\times01\times2E\times09\times00\times28\times01\times2F\times09\times00\times28$ "\x01\x30\x0A\x00\x3C\x01\x31\x0A\x00\x30\x01\x32\x0A\x00\x28\x01\x33\x0A\x00\x46\x01\x34" $"\x09\x00\x28\x01\x35\x0A\x00\x39\x01\x36\x0A\x00\x30\x01\x37\x0A\x00\x36\x01\x38\x0A\x00"$ $"\x30\x01\x39\x0A\x00\x30\x01\x3A\x0A\x00\x31\x01\x3B\x0A\x00\x39\x01\x3C\x0A\x00\x46\x01"$. $"\times3D\times09\times00\times28\times01\times3E\times01\times3E\times01\times3F\times00\times46\times01\times3F\times00\times43\times01\times40\times00$ $"\x00\x2A\x01\x41\x0A\x00\x46\x01\x41\x0A\x00\x28\x01\x42\x0A\x00\x43\x01\x43\x0A\x00\x43"$ "\x01\x44\x0A\x00\x46\x01\x45\x09\x00\x28\x01\x46\x09\x00\x28\x01\x47\x09\x00\x28\x01\x48" "\x35\x01\x4D\x09\x00\x35\x01\x4E\x09\x00\x2F\x01\x4F\x09\x00\x32\x01\x50\x0C\x01\x87\x01" "\x60\x0C\x01\x87\x01\x67\x0C\x01\x87\x01\x68\x0C\x01\x87\x01\x69\x0C\x01\x87\x01\x6F\x0C" "\x01\x87\x01\x75\x0C\x01\x87\x01\x76\x0C\x01\x87\x01\x79\x0C\x01\x87\x01\x7F\x0C\x01\x87" "\x01\x82\x0C\x01\x87\x01\x83\x0C\x01\x87\x01\x84\x0C\x01\xB5\x01\x97\x0C\x01\xB6\x01\x6C" - $"\x0C\x01\x86\x01\x6E\x0C\x01\x86\x01\x7F\x0C\x01\x87\x01\x74\x0C\x01\x88\x01\x63\x0C\x01"$ "\xB8\x01\x7E\x0C\x01\xBA\x01\xA5\x0C\x01\xBB\x01\x9C\x0C\x01\xBD\x01\xA5\x0C\x01\xBF\x01" $\label{label} $$ \x 0 C \times 0 1 \times 0 0 \times 0 1 \times 0 0 \times 0 1 \times 0 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 0 \times 0 0 \times 0 0 \times 0 \times 0 0 \times 0 0 \times 0 \times$ $"\x01\xC5\x01\xA3\x0C\x01\xC6\x01\xC5\x0C\x01\xC7\x01\xC7\x01\xC9\x01\xC9\x01\xC4"$ "\x01\x97\x0C\x01\xCB\x01\x60\x0C\x01\xCC\x01\x60\x0C\x01\xCE\x01\x9C" $"\x0C\x01\xD0\x01\xA5\x0C\x01\xD2\x01\xD2\x01\xD3\x01\xA5\x0C\x01\xD4\x01\xB4\x0C\x01"$ "\xD5\x01\x9B\x0C\x01\xD6\x01\xB4\x0C\x01\xD7\x01\x97\x0C\x01\xDB\x01\xA5\x0C\x01\xDC\x01" "\x01\xE1\x01\x59\x0C\x01\xE2\x01\x57\x0C\x01\xE3\x01\x5A\x0C\x01\xE4\x01\x5B\x0C\x01\xE5" "\x01\x81\x0C\x01\xE6\x01\x5D\x0C\x01\xE7\x01\x5D\x0C\x01\xE8\x01\x7D\x0C\x01\xEA\x01\x56" "\x0C\x01\xEB\x01\x55\x0C\x01\xEC\x01\x55\x0C\x01\xEC\x01\x58\x0C\x01\xED\x01\x5C\x0C\x01"

"\xEE\x01\x5D\x0C\x01\xEF\x01\x9C\x0C\x01\xF0\x01\x97\x0C\x01\xF1\x01\x97\x0C\x01\xF2\x01" "\x97\x0C\x01\xF3\x01\x97\x0C\x01\xF4\x01\x9E\x0C\x01\xF5\x01\x97\x0C\x01\xF6\x01\xA0\x0C" "\x01\x60\x0C\x01\xFD\x01\x55\x0C\x01\xFE\x01\x60\x0C\x01\xFF\x01\x60\x0C\x02\x22\x01\xA2" "\x27\x01\x9E\x0C\x02\x28\x01\x97\x0C\x02\x29\x01\x97\x0C\x02\x2A\x01\x5F\x0C\x02\x2B\x01" $\label{label} $$ \xA4\x0C\x02\x2D\x01\x7A\x0C\x02\x2E\x01\x7F\x0C\x02\x2F\x01\x97\x0C\x02\x30\x01\xA2\x0C"$$ $"\x01\x01\x02\x38\x01\x64\x00\x02\x39\x01\x6B\x00\x02\x3A\x01\x6D\x00\x02\x3B\x01\x70"$ $"\x0C\x02\x3C\x01\x6B\x0C\x02\x3D\x01\x72\x0C\x02\x3E\x01\x7F\x0C\x02\x3F\x01\x6A\x0C\x02"$ $"\x40\x01\x9A\x0C\x02\x41\x01\x60\x0C\x02\x42\x01\x80\x0C\x02\x43\x01\x60\x0C\x02\x44\x01"$ $"\x7B\x0C\x02\x45\x01\x62\x0C\x02\x45\x01\x66\x0C\x02\x46\x01\x60\x0C\x02\x47\x01\xB3\x0C"$ $"\x01\x60\x0C\x02\x4E\x01\x77\x0C\x02\x4F\x01\x8D\x0C\x02\x50\x01\x8D\x0C\x02\x51\x01\x9C"$ $"\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x68\x65\x63\x6B\x62\x6F\x78\x3B\x01\x00\x12\x28\x29\"$ "\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6C\x6F\x72\x3B\x01\x00\x16\x28\x29\x4C" $"\x61\x61\x76\x61\x2F\x61\x77\x74\x2F\x44\x69\x65\x6E\x73\x69\x6F\x6E\x38\x01\x00\x11"$ "\x28\x29\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x46\x6E\x74\x3B\x01\x00\x15\x28\x29". $"\x28\x29\x4C\x6A\x61\x76\x61\x2F\x69\x6F\x2F\x49\x6E\x70\x75\x74\x53\x74\x72\x65\x61\x6D"$ $"\x3B\x01\x00\x14\x28\x29\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x4F\x62\x6A\x65\x63"$ $"\x74\x38\x01\x00\x14\x28\x29\x4C\x64\x61\x76\x61\x2F\x6C\x61\x65\x74\x72\x69"$ $"\x6E\x67\x3B\x01\x00\x10\x28\x29\x4C\x6A\x61\x76\x61\x2F\x6E\x65\x74\x2F\x55\x52\x4C\x3B"$ "\x01\x00\x1A\x28\x29\x4C\x6A\x61\x76\x61\x2F\x6E\x65\x74\x2F\x55\x52\x4C\x43\x6F\x6E\x6E" $"\x65\x63\x74\x69\x6F\x6E\x3B\x01\x00\x03\x29\x29\x56\x01\x00\x16\x28\x29\x5B\x4C\x6A"$ "\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x15\x28\x49\x29" · "\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x1B\x28" "\x49\x29\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x42\x75\x66" "\x66\x65\x72\x3B\x01\x00\x04\x28\x49\x29\x56\x01\x00\x14\x28\x49\x49\x29\x4C\x6A\x61\x76" "\x61\x2F\x61\x77\x74\x2F\x49\x6D\x61\x67\x65\x3B\x01\x00\x16\x28\x49\x49\x29\x4C\x6A\x61" "\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x05\x28\x49\x49\x29" "\x04\x28\x4A\x29\x56\x01\x00\x13\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6C" "\x6F\x72\x3B\x29\x56\x01\x00\x2A\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6D" "\x70\x6F\x6E\x65\x6E\x74\x3B\x29\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6D\x70" "\x6F\x6E\x65\x6E\x74\x3B\x01\x00\x34\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F"

"\x6D\x70\x6F\x6E\x65\x6E\x74\x3B\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x47\x72\x69\x64" "\x42\x61\x67\x43\x6F\x6E\x73\x74\x72\x61\x69\x6E\x74\x73\x3B\x29\x56\x01\x00\x29\x28\x4C" "\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6D\x70\x6F\x6E\x65\x6E\x74\x3B\x4C\x6A\x61" "\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x4F\x62\x6A\x65\x63\x74\x3B\x29\x56\x01\x00\x17\x28\x4C" "\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x44\x69\x6D\x65\x6E\x73\x69\x6F\x6E\x3B\x29\x56\x01" "\x00\x12\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x46\x6F\x6E\x74\x3B\x29\x56\x01\x00" "\x16\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x47\x72\x61\x70\x68\x69\x63\x73\x3B\x29" "\x56\x01\x00\x1B\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x4C\x61\x79\x6F\x75\x74\x4D" "\x61\x6E\x61\x67\x65\x72\x3B\x29\x56\x01\x00\x1D\x28\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74" "\x2F\x65\x76\x65\x6E\x74\x2F\x49\x74\x65\x6D\x45\x76\x65\x6E\x74\x3B\x29\x56\x01\x00\x20" "\x69\x73\x74\x65\x6E\x65\x72\x3B\x29\x56\x01\x00\x18\x28\x4C\x6A\x61\x76\x61\x2F\x69\x6F" "\x2F\x49\x6E\x70\x75\x74\x53\x74\x52\x65\x61\x6D\x3B\x29\x56\x01\x00\x13\x28\x4C\x6A\x61" · "\x76\x61\x2F\x69\x6F\x2F\x52\x65\x61\x64\x65\x72\x3B\x29\x56\x01\x00\x26\x28\x4C\x6A\x61" "\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x4F\x62\x6A\x65\x63\x74\x3B\x29\x4C\x6A\x61\x76\x61\x2F" "\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x15\x28\x4C\x6A\x61\x76\x61\x2F\" $"\x6C\x61\x6E\x67\x2F\x4F\x62\x6A\x65\x63\x74\x3B\x29\x5A\x01\x00\x17\x28\x4C\x6A\x61\x76"$ $"\times61\times2F\times6C\times61\times6F\times67\times2F\times52\times75\times6E\times6E\times61\times62\times65\times3B\times29\times56\times01\times00\times15\times28$ "\x4C\x6a\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x49\x01\x00" "\x24\x28\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x4C" "\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6C\x6F\x72\x3B\x01\x00\x27\x28\x4C\x6A\x61" "\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x4C\x6A\x61\x76\x61\x2F" "\x6C\x61\x6E\x67\x2F\x49\x6E\x74\x65\x67\x65\x72\x3B\x01\x00\x26\x28\x4C\x6A\x61\x76\x61" "\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x4C\x6A\x61\x76\x61\x2F\x6C\x61" "\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x2C\x28\x4C\x6A\x61\x76\x61\x2F\x6C\x61" "\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F" "\x53\x74\x72\x69\x6E\x67\x42\x75\x66\x66\x65\x72\x3B\x01\x00\x15\x28\x4C\x6A\x61\x76\x61" "\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x56\x01\x00\x15\x28\x4C\x6A\x61" "\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29\x5A\x01\x00\x16\x28\x4C" "\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x49\x29\x49\x01\x00" "\x17\x28\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x49\x49\" "\x29\x56\x01\x00\x2E\x28\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E" "\x67\x3B\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x68\x65\x63\x6B\x62\x6F\x78\x47\x72" $"\x6F\x75\x70\x3B\x5A\x29\x56\x01\x00\x23\x28\x4C\x6A\x61\x76\x61\x2F\x6E\x65\x74\x2F\x55"$ "\x52\x4C\x3B\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x29" "\x56\x01\x00\x06\x30\x30\x30\x30\x30\x30\x01\x00\x02\x30\x78\x01\x00\x06\x3C\x69\x6E\x69" "\x74\x3E\x01\x00\x01\x42\x01\x00\x21\x42\x61\x63\x6B\x67\x72\x6F\x75\x6E\x64\x20\x63\x6F" "\x6C\x6F\x72\x2C\x20\x66\x6F\x72\x60\x61\x74\x20\x22\x72\x72\x67\x67\x62\x62\x22\x01\x00"

 $"\x06\x43\x30\x43\x30\x43\x30\x01\x00\x04\x43\x6F\x64\x65\x01\x00\x00\x43\x6F\x6E\x73\x74"$ $"\x20\x44\x69\x73\x70\x6C\x61\x79\x3A\x01\x00\x0D\x44\x65\x66\x61\x75\x6C\x74\x20\x6C\x61"$ $"\x62\x65\x6C\x01\x00\x24\x44\x65\x6C\x61\x79\x20\x62\x65\x74\x77\x65\x65\x65\x75\x70"$ $"\x04\x45\x61\x73\x74\x01\x00\x22\x45\x72\x72\x6F\x72\x20\x61\x6C\x6F\x63\x61\x74\x69"$ $"\x6F\x72\x65\x67\x72\x6F\x75\x6E\x64\x20\x65\x6F\x72\x2C\x20\x66\x6F\x72\x6D\x61"$ $"\x61\x62\x65\x6C\x20\x73\x74\x72\x69\x6E\x67\x20\x74\x6F\x20\x65\x20\x64\x69\x73\x70"$ $"\x65\x01\x00\x18\x4C\x64\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x68\x65\x63\x68\x62\x6F\x78"$ "\x47\x72\x6F\x75\x70\x3B\x01\x00\x11\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x68\x6F" - $"\x69\x63\x65\x3B\x01\x00\x10\x4C\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6C\x6F\x72"$ $"\x3B\x01\x00\x14\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x44\x69\x6D\x65\x6E\x73\x69\x6F"$ $"\times13\times4C\times6A\times61\times76\times61\times2F\times61\times77\times74\times2F\times47\times72\times61\times70\times68\times69\times63\times73\times3B\times01\times00"$ $"\x10\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x49\x61\x67\x65\x3B\x01\x00\x11\x4C\x6A"$ $"\x61\x76\x61\x2F\x61\x77\x74\x2F\x49\x6E\x73\x3B\x01\x00\x10\x4C\x6A\x61\x76"$ $"\x61\x2F\x61\x77\x74\x2F\x61\x62\x65\x6C\x3B\x01\x00\x10\x4C\x6A\x61\x76\x61\x2F\x61"$ $"\x77\x74\x2F\x50\x61\x6E\x65\x6C\x3B\x01\x00\x15\x4C\x6A\x61\x76\x61\x2F\x69\x6F\x2F\x50"$ $\x^72\x69\x6E\x74\x53\x74\x72\x65\x61\x6D\x3B\x01\x00\x12\x4C\x6A\x61\x76\x61\x2F\x6C\x61$ "\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x12\x4C\x6A\x61\x76\x61\x2F\x6C\x61\x6E" "\x67\x2F\x54\x68\x72\x65\x61\x64\x3B\x01\x00\x0E\x4C\x6F\x63\x61\x6C\x56\x61\x72\x69\x61" "\x20\x6F\x66\x20\x66\x6E\x61\x6D\x65\x58\x2C\x20\x66\x6C\x61\x62\x65\x6C\x58\x20\x61" "\x72\x61\x6D\x73\x01\x00\x16\x4E\x75\x6D\x62\x65\x72\x20\x6F\x66\x20\x74\x65\x74\x20" "\x63\x6F\x6C\x75\x6D\x6E\x73\x01\x00\x13\x4E\x75\x6D\x62\x65\x72\x20\x6F\x66\x20\x74\x65" "\x78\x74\x20\x72\x6F\x77\x73\x01\x00\x04\x53\x6C\x6F\x77\x01\x00\x0A\x53\x6F\x75\x72\x63" "\x65\x46\x69\x6C\x65\x01\x00\x06\x53\x74\x72\x69\x6E\x67\x01\x00\x07\x55\x52\x5F\x47\x72" "\x69\x64\x01\x00\x0C\x55\x52\x5F\x47\x72\x69\x64\x2E\x6A\x61\x76\x61\x01\x00\x0D\x55\x70" "\x64\x61\x74\x65\x20\x53\x70\x65\x65\x64\x3A\x01\x00\x04\x57\x65\x73\x74\x01\x00\x11\x5B" "\x4C\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x4C\x61\x62\x65\x6C\x3B\x01\x00\x13\x5B\x4C\x6A" "\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x53\x74\x72\x69\x6E\x67\x3B\x01\x00\x0A\x61\x63\x74" "\x69\x76\x65\x46\x69\x6C\x65\x01\x00\x03\x61\x64\x64\x01\x00\x0F\x61\x64\x64\x49\x74\x65" "\x61\x63\x6B\x67\x72\x6F\x75\x6E\x64\x01\x00\x0F\x62\x61\x63\x6B\x67\x72\x6F\x75\x6E\x64"

"\x6C\x65\x01\x00\x0B\x62\x67\x66\x69\x6C\x65\x50\x61\x72\x61\x00\x01\x00\x0A\x62\x67\x66" $"\x69\x6C\x65\x61\x6D\x65\x01\x00\x07\x62\x67\x6C\x69\x6E\x65\x73\x01\x00\x05\x62\x6C\$ $"\x61\x63\x6b\x01\x00\x09\x63\x6c\x65\x61\x72\x52\x65\x63\x74\x01\x00\x05\x63\x6c\x6F\x73"$ "\x65\x01\x00\x04\x63\x6F\x6C\x73\x01\x00\x04\x63\x6F\x6C\x73\x50\x61\x72\x61\x60\x01\x00" $"\x08\x63\x6F\x6E\x74\x72\x6F\x6C\x73\x01\x00\x08\x63\x72\x65\x61\x74\x65\x49\x6D\x61\x67"$ $"\times65\times01\times00\times66\times64\times65\times63\times6F\times64\times65\times01\times00\times05\times64\times65\times61\times79\times01\times00\times064$ $"\x65\x6C\x61\x79\x50\x61\x72\x61\x6D\x01\x00\x09\x64\x65\x6C\x61\x79\x54\x69\x6D\x65\x01"$ "\x00\x07\x64\x65\x73\x74\x72\x6F\x79\x01\x00\x08\x64\x6F\x4C\x61\x79\x6F\x75\x74\x01\x00" "\x06\x65\x71\x75\x61\x6C\x73\x01\x00\x07\x66\x67\x43\x6F\x6C\x6F\x72\x01\x00\x06\x66\x67" "\x66\x69\x6C\x65\x01\x00\x0B\x66\x67\x66\x69\x6C\x65\x50\x61\x72\x61\x60\x01\x00\x0A\x66" "\x67\x66\x69\x6C\x65\x6E\x61\x6D\x65\x01\x00\x09\x66\x69\x6C\x65\x43\x6F\x75\x6E\x74\x01" "\x00\x0E\x66\x69\x6C\x65\x43\x6F\x75\x6E\x74\x50\x61\x72\x61\x61\x00\x01\x00\x0A\x66\x69\x6C" "\x65\x4C\x61\x62\x65\x6C\x73\x01\x00\x08\x66\x65\x4C\x65\x4C\x69\x73\x74\x01\x00\x09\x66" "\x69\x6C\x65\x4E\x61\x6D\x65\x73\x01\x00\x04\x66\x69\x6C\x61\x00\x06\x66\x66\x61\x62" $"\x65\x6C\x01\x00\x05\x66\x6E\x61\x60\x65\x01\x00\x0A\x66\x6F\x72\x65\x67\x72\x6F\x75\x6E"$ $"\x64\x01\x00\x0F\x66\x6F\x72\x65\x67\x72\x6F\x75\x6E\x64\x50\x61\x72\x61\x6D\x01\x00\x02\"$ "\x67\x63\x01\x00\x0C\x67\x65\x74\x41\x6C\x69\x67\x6E\x6D\x65\x6E\x74\x01\x00\x0D\x67\x65" $"\x74\x42\x61\x63\x6B\x67\x72\x6F\x75\x6E\x64\x01\x00\x07\x67\x65\x74\x44\x61\x74\x61\x01"$ "\x00\x0F\x67\x65\x74\x44\x6F\x63\x75\x6D\x65\x6E\x74\x42\x61\x73\x65\x01\x00\x07\x67\x65" "\x74\x46\x6F\x6E\x74\x01\x00\x0D\x67\x65\x74\x46\x6F\x72\x65\x75\x6E\x75\x6E\x64\x01" "\x00\x0B\x67\x65\x74\x47\x72\x61\x70\x68\x69\x63\x73\x01\x00\x0E\x67\x65\x74\x49\x6E\x70" $"\x75\x74\x53\x74\x72\x65\x61\x60\x01\x00\x08\x67\x65\x74\x49\x6E\x74\x50\x61\x72\x61\x60"$ "\x01\x00\x08\x67\x65\x74\x4C\x61\x62\x65\x6C\x01\x00\x07\x67\x65\x74\x4E\x61\x6D\x65\x01" "\x00\x0C\x67\x65\x74\x50\x61\x72\x61\x6D\x65\x74\x65\x72\x01\x00\x10\x67\x65\x74\x50\x61" "\x72\x61\x6D\x65\x74\x65\x72\x49\x6E\x66\x6F\x01\x00\x13\x67\x65\x74\x53\x65\x6C\x65\x63" "\x74\x65\x64\x49\x6E\x64\x65\x78\x01\x00\x07\x67\x65\x74\x53\x69\x7A\x65\x01\x00\x09\x67" $"\x65\x74\x53\x6F\x75\x72\x63\x65\x01\x00\x07\x67\x65\x74\x54\x65\x78\x74\x01\x00\x04\x67"$ "\x72\x61\x79\x01\x00\x0A\x67\x72\x69\x64\x68\x65\x69\x64\x68\x74\x01\x00\x09\x67\x72\x69" $"\x64\x77\x69\x64\x74\x68\x01\x00\x05\x67\x72\x69\x64\x78\x01\x00\x05\x67\x72\x69\x64\x79"$ $"\x01\x00\x0B\x68\x65\x61\x64\x69\x6E\x67\x6E\x74\x01\x00\x06\x68\x65\x69\x67\x68"$ $\x01\x01\x00\x0B\x69\x61\x67\x65\x42\x75\x66\x65\x72\x01\x00\x09\x69\x60\x61\x67\$ "\x65\x53\x69\x7A\x65\x01\x00\x04\x69\x6E\x69\x74\x01\x00\x08\x6E\x69\x74\x46\x6F\x72" $"\x6D\x01\x00\x69\x6E\x70\x75\x74\x46\x69\x6E\x65\x4E\x61\x6D\x65\x01\x00\x06\x69\x6E"$ $"\x73\x65\x74\x73\x01\x00\x0F\x69\x6E\x73\x74\x61\x6C\x43\x6F\x6E\x74\x72\x6F\x6C\x73"$ "\x01\x00\x08\x69\x6E\x74\x56\x61\x6C\x75\x65\x01\x00\x09\x6E\x74\x65\x72\x72\x75\x70" "\x74\x01\x00\x0A\x69\x6E\x76\x61\x6C\x69\x64\x61\x74\x65\x01\x00\x10\x69\x74\x65\x6D\x53" "\x74\x61\x74\x65\x43\x68\x61\x6E\x67\x65\x64\x01\x00\x12\x6A\x61\x76\x61\x2F\x61\x70\x70" $"\x6C\x65\x74\x2F\x41\x70\x70\x6C\x65\x74\x01\x00\x15\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F"$ "\x43\x68\x65\x63\x6B\x62\x6F\x78\x47\x72\x6F\x75\x70\x01\x00\x0F\x6A\x61\x76\x61\x2F\x61" "\x77\x74\x2F\x43\x68\x6F\x69\x63\x65\x01\x00\x0E\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43" $"\x6F\x6C\x6F\x72\x01\x00\x12\x6A\x61\x76\x61\x77\x74\x2F\x43\x6F\x6D\x70\x6F\x6E"$ $"\x65\x6E\x74\x01\x00\x12\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x43\x6F\x6E\x74\x61\x69\x6E"$ $"\x65\x72\x01\x00\x12\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x44\x69\x6D\x65\x6E\x73\x69\x6F"$ "\x6E\x01\x00\x0D\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x46\x6E\x74\x01\x00\x11\x6A\x61" $"\x76\x61\x2F\x61\x77\x74\x2F\x47\x72\x61\x70\x68\x69\x63\x73\x01\x00\x1B\x6A\x61\x76\x61"$ "\x2F\x61\x77\x74\x2F\x47\x72\x69\x64\x42\x61\x67\x43\x6F\x6E\x73\x74\x72\x61\x69\x6E\x74" "\x73\x01\x00\x16\x64\x61\x76\x61\x2F\x61\x77\x74\x2F\x47\x72\x69\x64\x42\x61\x67\x4C\x61" $"\x79\x6F\x75\x74\x01\x00\x0E\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x49\x6D\x61\x67\x65\x01"$ "\x00\x0F\x6A\x61\x76\x61\x2F\x61\x77\x74\x2F\x49\x6E\x73\x65\x74\x73\x01\x00\x0E\x6A\x61" $"\x76\x61\x2F\x61\x77\x74\x2F\x4C\x61\x62\x65\x6C\x01\x00\x0E\x6A\x61\x76\x61\x2F\x61\x77"$ $"\x6E\x74\x2F\x49\x74\x65\x6E\x65\x72\x01\x00\x16\x6A\x61\x76\x61"$ "\x2F\x69\x6F\x2F\x42\x75\x66\x66\x65\x72\x65\x64\x52\x65\x61\x64\x65\x72\x01\x00\x19\x6A" "\x61\x76\x61\x2F\x69\x6F\x2F\x49\x6E\x70\x75\x74\x53\x74\x72\x65\x61\x6D\x52\x65\x61\x64" "\x65\x72\x01\x00\x13\x6A\x61\x76\x61\x2F\x69\x6F\x2F\x50\x72\x69\x6E\x74\x53\x74\x72\x65" $"\x61\x60\x01\x00\x13\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x45\x78\x65\x70\x74\x69$ " $"\x6F\x6E\x01\x00\x11\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x49\x6E\x74\x65\x67\x72"$ $"\x01\x00\x1E\x6A\x61\x2F\x6G\x61\x2E\x6A\x61\x2F\x6G\x61\x6E\x74\x65\x72\x72\x75\x70\x74"$ $"\x65\x64\x45\x78\x65\x70\x74\x69\x6F\x6E\x01\x00\x12\x6A\x61\x76\x6C\x61\x6E"$ "\x2F\x53\x74\x72\x69\x6E\x67\x01\x00\x16\x6A\x61\x76\x61\x2F\x6C\x61\x6F\x67\x2F\x53\x74" "\x72\x69\x6E\x67\x42\x75\x66\x66\x65\x72\x01\x00\x10\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67" "\x2F\x53\x79\x73\x74\x65\x6D\x01\x00\x10\x6A\x61\x76\x61\x2F\x6C\x61\x6E\x67\x2F\x54\x68" "\x72\x65\x61\x64\x01\x00\x0C\x6A\x61\x76\x61\x2F\x6E\x65\x74\x2F\x55\x52\x4C\x01\x00\x16" "\x6A\x61\x76\x61\x2F\x6E\x65\x74\x2F\x55\x52\x4C\x43\x6F\x6E\x65\x63\x74\x69\x6F\x6E" "\x01\x00\x15\x6A\x61\x76\x61\x2F\x75\x74\x69\x6C\x2F\x45\x76\x6E\x74\x4F\x62\x6A\x65" "\x63\x74\x01\x00\x05\x6C\x61\x62\x65\x6C\x01\x00\x06\x6C\x61\x62\x65\x6C\x31\x01\x00\x0A" "\x6C\x61\x62\x65\x6C\x50\x61\x72\x61\x6D\x01\x00\x06\x6C\x65\x6E\x67\x74\x68\x01\x00\x00" "\x6D\x61\x69\x6E\x54\x69\x74\x6C\x65\x46\x6F\x6E\x74\x01\x00\x08\x6E\x42\x67\x6C\x69\x6E" "\x65\x73\x01\x00\x0A\x6E\x6F\x72\x6D\x61\x6C\x46\x6F\x6E\x74\x01\x00\x07\x6E\x75\x6D\x43" "\x6F\x6C\x73\x01\x00\x07\x6E\x75\x6D\x52\x6F\x77\x73\x01\x00\x0E\x6F\x70\x65\x6E\x43\x6F" "\x6E\x6E\x65\x63\x74\x69\x6F\x6E\x01\x00\x03\x6F\x75\x74\x01\x00\x05\x70\x61\x69\x6E\x74"

"\x01\x00\x08\x70\x61\x72\x73\x65\x49\x6E\x74\x01\x00\x07\x70\x72\x69\x6E\x74\x6C\x6E\x01" "\x00\x0D\x70\x72\x6F\x67\x72\x65\x73\x73\x43\x6F\x75\x6E\x74\x01\x00\x0D\x70\x72\x6F\x6F\x6F "\x72\x65\x73\x73\x4C\x61\x62\x65\x6C\x01\x00\x08\x72\x65\x61\x64\x4C\x69\x6E\x65\x01\x00" "\x07\x72\x65\x70\x61\x69\x6E\x74\x01\x00\x0B\x72\x65\x73\x69\x7A\x65\x49\x61\x61\x67\x65" "\x01\x00\x06\x72\x65\x73\x75\x6D\x65\x01\x00\x04\x72\x6F\x77\x73\x01\x00\x09\x72\x6F\x77" "\x73\x50\x61\x72\x61\x6D\x01\x00\x03\x72\x75\x6E\x01\x00\x02\x73\x65\x74\x41\x6C\x69\x67" "\x6E\x6D\x65\x6E\x74\x01\x00\x0D\x73\x65\x74\x42\x61\x63\x6B\x67\x72\x6F\x75\x6E\x64\x01" "\x00\x0E\x73\x65\x74\x43\x6F\x6E\x73\x74\x72\x61\x69\x6E\x74\x73\x01\x00\x07\x73\x65\x74" "\x46\x6F\x6E\x74\x01\x00\x0D\x73\x65\x74\x46\x6F\x72\x65\x67\x72\x6F\x75\x6E\x64\x01\x00" "\x09\x73\x65\x74\x4C\x61\x79\x6F\x75\x74\x01\x00\x07\x73\x65\x74\x54\x65\x78\x74\x01\x00" "\x05\x73\x6C\x65\x65\x70\x01\x00\x0B\x73\x70\x65\x65\x64\x53\x65\x6C\x65\x63\x74\x01\x00" "\x05\x73\x74\x61\x72\x74\x01\x00\x0A\x73\x74\x61\x72\x74\x57\x69\x74\x68\x01\x00\x04" "\x73\x74\x6F\x70\x01\x00\x0D\x73\x74\x72\x69\x6E\x67\x54\x6F\x43\x6F\x6C\x6F\x72\x01\x00" $"\x09\x73\x75\x62\x73\x74\x72\x69\x6E\x67\x01\x00\x07\x73\x75\x73\x70\x65\x6E\x64\x01\x00"$ "\x07\x74\x68\x65\x54\x65\x78\x74\x01\x00\x05\x74\x69\x6D\x65\x72\x01\x00\x08\x74\x69\x74" "\x6C\x65\x42\x61\x72\x01\x00\x09\x74\x69\x74\x6C\x65\x46\x6F\x6E\x74\x01\x00\x08\x74\x6F" $"\times 53 \times 74 \times 72 \times 69 \times 62 \times 67 \times 01 \times 00 \times 06 \times 75 \times 70 \times 64 \times 61 \times 74 \times 65 \times 01 \times 00 \times 75 \times 73 \times 65 \times 50$ "\x61\x67\x65\x50\x61\x72\x61\x6D\x73\x01\x00\x07\x76\x61\x65\x4F\x66\x01\x00\x07" "\x77\x65\x69\x67\x68\x74\x78\x01\x00\x07\x77\x65\x69\x67\x68\x74\x79\x01\x00\x05\x77\x68" "\x69\x74\x65\x01\x00\x05\x77\x69\x64\x74\x68\x01\x00\x01\x7E\x00\x21\x00\x28\x00\x2A\x00" "\x02\x00\x3B\x00\x42\x00\x27\x00\x02\x02\x48\x01\xA6\x00\x00\x00\x02\x01\xF6\x01\xA0\x00" $"\x00\x00\x02\x01\x9F\x00\x00\x00\x00\x01\xF7\x01\x9D\x00\x00\x02\x02\x01"$ "\xA5\x00\x01\x01\x8C\x00\x00\x00\x00\x00\x24\x00\x12\x01\xBA\x01\xA5\x00\x01\x01\x8C\x00" "\x01\x01\x8C\x00\x00\x00\x02\x00\x1F\x00\x12\x01\xC9\x01\xA5\x00\x01\x8C\x00\x00\x00" "\xC4\x01\xA5\x00\x01\x01\x8C\x00\x00\x00\x02\x00\x1D\x00\x12\x01\xD3\x01\xA5\x00\x01\x01" "\x8C\x00\x00\x00\x02\x00\x20\x00\x02\x01\xBB\x01\x9C\x00\x00\x02\x01\xCE\x01\x9C\x00" "\x00\x00\x02\x01\xBE\x01\xA5\x00\x00\x00\x02\x01\xD1\x01\x01\xA5\x00\x00\x00\x01\xCA\x01" "\x00\x00\x02\x02\x25\x01\x9E\x00\x00\x00\x02\x01\xF4\x01\x9E\x00\x00\x00\x02\x4A\x01" $"\x00\x01\xD5\x01\x9B\x00\x00\x00\x00\x02\x30\x01\xA2\x00\x00\x00\x02\x2F\x01\x97\x00"$ "\x00\x00\x01\xBF\x01\xB4\x00\x00\x00\x02\x26\x01\x97\x00\x00\x01\x00\x01\x01\x01"

"\xF8\x01\x60\x00\x01\x01\x8B\x00\x00\x00\x39\x00\x01\x00\x01\x00\x00\x11\x2A\xB6\x00" "\x9D\x2A\xB7\x00\xCE\x2A\xB6\x00\xA0\x2A\xB6\x00\xA3\xB1\x00\x00\x01\x01\x99\x00\x00" $"\x00\x16\x00\x05\x00\x00\x18\x00\x04\x00\x1A\x00\x08\x00\x1B\x00\x00\x1C\x0$ "\x00\x16\x00\x00\x01\xE5\x01\x81\x00\x01\x01\x8B\x00\x00\x00\x4F\x00\x02\x00\x06\x00\x00" "\x00\x1F\x2A\x2B\xB6\x00\x8D\x4E\x1C\x36\x04\x2D\xC6\x00\x12\x2D\xB8\x00\xAE\x36\x05\x15" $\x05\x9E\x00\x07\x15\x05\x36\x04\x15\x04\xAC\x00\x00\x00\x01\x01\x99\x00\x00\x00\x1E\x00$ $"\x18\x00\x3E\x00\x1C\x00\x40\x00\x02\x02\x4D\x01\x60\x00\x01\x01\x8B\x00\x00\x01\xE2\x00"$ "\x05\x00\x0A\x00\x01\x42\x12\x0B\x4C\x12\x09\x4D\x12\x05\x4E\x2A\x12\x24\xB6\x00\x8D" "\x3A\x04\x2A\x12\x1B\xB6\x00\x8D\x3A\x05\x2A\x12\x23\xB6\x00\x8D\x3A\x06\x19\x04\xC6\x00" "\x0D\x19\x05\xC6\x00\x08\x19\x06\xC7\x00\x0C\x2B\x3A\x04\x2C\x3A\x05\x2D\x3A\x06\x2A\x2A\x2A\x "\x12\x1E\x11\x03\xE8\xB6\x00\x8A\xB5\x00\x71\x2A\x2A\x12\x26\x10\x0A\xB6\x00\x8A\xB5\x00" "\xAB\x2A\x2A\x12\x1D\x04\xB6\x00\x8A\xB5\x00\xAA\x2A\x2A\x12\x20\x03\xB6\x00\x8A\xB5\x00" $"\x7A\x2A\x2A\xB4\x00\x7A\xBD\x00\x43\xB5\x00\x7C\x2A\xB4\x00\x7A\xBD\x00\x43\xB5\x00"$ "\x7E\x03\x36\x07\xA7\x00\x60\xBB\x00\x44\x59\x12\xB7\x00\x5A\x15\x07\x04\x60\xB6\x00" $"\x63\xB6\x00\xCD\x3A\x08\x2A\x19\x08\xB6\x00\x3A\x09\x2A\xB4\x00\x7C\x15\x07\xBB\x00"$ $"\x00\x63\xB6\x00\xCD\x3A\x08\x2A\x19\x08\xB6\x00\x8D\x3A\x09\x2A\xB4\x00\x7E\x15\x07\xBB"$ "\xB4\x00\x7A\x9E\x00\x10\x2A\x2A\xB4\x00\x7E\x2A\xB4\x00\x5E\x32\xB5\x00\x9E\x2A\xB4\x00" $"\x04\x19\x04\x06\x00\x66\x2A\x2A\x19\x05\x87\x00"$ $"\xC5\xB5\xO0\x78\x2A\xB4\xO0\xA4\x2A\xB4\xO0\xB6\xB6\xO0\xB8\x2A\xB4\xO0\xA4\x2A\xB4\xO0"$ "\x78\xB6\x00\xBB\x2A\x2A\xB4\x00\x66\xB6\x00\xB8\x2A\x2A\xB4\x00\x78\xB6\x00\xBB\xB1\x00" $"\x4B\x00\x09\x00\x56\x00\x11\x00\x57\x00\x19\x00\x58\x00\x21\x00\x55\x00\x2B\x00\"$ "\x68\x00\x5D\x00\x6B\x00\x68\x00\x6C\x00\x73\x00\x6D\x00\x7E\x00\x6E\x00\x84\x00\x72\x00" "\x6E\x00\xEA\x00\x79\x00\xF1\x00\x7A\x00\xFE\x00\x7F\x01\x07\x00\x80\x01\x11\x00\x81\x01" $"\x18\x00\x82\x01\x26\x00\x83\x01\x31\x00\x84\x01\x39\x00\x85\x01\x41\x00\x47\x00\x02\x02\"$ "\x59\x12\x06\xB7\x00\x5A\x2B\x03\x05\xB6\x00\xC7\xB6\x00\x64\xB6\x00\xCD\xB8\x00\x6F\xB6" "\x00\xA1\x3D\xBB\x00\x44\x59\x12\x06\xB7\x00\x5A\x2B\x05\x07\xB6\x00\xC7\xB6\x00\x64\xB6". "\x00\xCD\xB8\x00\x6F\xB6\x00\xA1\x3E\xBB\x00\x44\x59\x12\x06\xB7\x00\x5A\x2B\x07\x10\x06" "\xB6\x00\xC7\xB6\x00\x64\xB6\x00\xCD\xB8\x00\x6F\xB6\x00\xA1\x36\x04\xBB\x00\x2F\x59\x1C" "\x91\x00\x1C\x00\x92\x00\x38\x00\x93\x00\x56\x00\x95\x00\x01\x01\xE9\x01\x61\x00\x01\x01" "\x8B\x00\x00\x00\xFB\x00\x07\x00\x02\x00\x00\x00\xA3\x10\x07\xBD\x00\x29\x59\x03\x06\xBD"

"\x00\x43\x59\x03\x12\x24\x53\x59\x04\x12\x18\x53\x59\x05\x12\x12\x53\x59\x04\x06\xBD" "\x00\x43\x59\x03\x12\x1B\x53\x59\x04\x12\x18\x53\x59\x05\x12\x08\x53\x53\x59\x06\xBD" -"\x00\x43\x59\x03\x12\x23\x53\x59\x04\x12\x18\x53\x59\x05\x12\x10\x53\x59\x06\x06\xBD" "\x00\x43\x59\x03\x12\x1E\x53\x59\x04\x12\x18\x53\x59\x05\x12\x0C\x53\x59\x07\x06\xBD" "\x00\x43\x59\x03\x12\x26\x53\x59\x04\x12\x18\x53\x59\x05\x12\x16\x53\x59\x08\x06\xBD" "\x00\x43\x59\x03\x12\x1D\x53\x59\x04\x12\x18\x53\x59\x05\x12\x15\x53\x59\x10\x06\x06" "\xBD\x00\x43\x59\x03\x12\x20\x53\x59\x04\x12\x18\x53\x59\x05\x12\x14\x53\x53\x4C\x2B\xB0" "\x00\x00\x00\x01\x01\x99\x00\x00\x46\x00\x11\x00\x00\x02\x00\x07\x00\xA3\x00\x1A" $"\x00\x32\x00\x1D\x00\x34\x00\x30\x00\x32\x00\x35\x00\x46\x00\x32\x00\x49\x00\x36"$ "\x00\x5C\x00\xA2\x00\x5F\x00\xA7\x00\x72\x00\xA2\x00\x75\x00\xA8\x00\x88\x00\xA2\x00\x8C" "\x00\xA9\x00\x9F\x00\xA2\x00\xA0\x00\xA1\x00\xA1\x00\xAB\x00\x00\x01\xF9\x01\x60\x00\x01" "\x01\x8B\x00\x00\x00\x3E\x00\x02\x00\x01\x00\x00\x01\x2A\x2A\x2A\xB4\x00\x66\xB6\x00\xB8" "\x2A\x2A\xB4\x00\x78\xB6\x00\xBB\x2A\xB4\x00\xA4\x12\x25\xB6\x00\xB1\x00\x00\x00\x00\x01" "\x01\x99\x00\x00\x00\x12\x00\x04\x00\x00\x00\xB5\x00\x08\x00\xB6\x00\x10\x00\xB7\x00\x19" "\x00\xB3\x00\x00\x01\xFC\x01\x60\x00\x01\x01\x8B\x00\x03\xDE\x00\x07\x00\x0A\x00\x00" "\x02\xAE\xBB\x00\x36\x59\xB7\x00\x4E\x4C\x2A\x2B\xB6\x00\xBE\xBB\x00\x35\x59\xB7\x00\x4D" "\x4D\x2C\xBB\x00\x38\x59\x04\x04\x04\x04\xB7\x00\x53\xB5\x00\x9F\x2C\x0F\xB5\x00\xD0\x2C" "\x0F\xB5\x00\xD1\x2C\x04\xB5\x00\x7F\x2C\x04\xB5\x00\x97\x2C\x04\xB5\x00\x98\x2C\x04\xB5" "\x00\x95\x2C\x2A\xB4\x00\xAA\xB5\x00\x96\x2B\x2A\xB4\x00\xCB\x2C\xB6\x00\xB9\x2A\xB4\" "\x00\xCB\xB6\x00\x5F\x57\x2A\x2A\xB6\x00\x86\xB5\x00\xA9\x2A\xBB\x00\x33\x59\x2A\xB4\x00" "\x2A\xBB\x00\x33\x59\x2A\xB4\x00\xA9\xB6\x00\x8C\x04\x2A\xB4\x00\xA9\xB6\x00\x90\xB7\x00" "\x5B\xB5\x00\x99\x2A\xBB\x00\x33\x59\x2A\xB4\x00\xA9\xB6\x00\x8C\x06\x2A\xB4\x00\xA9\xB6" "\x00\x90\xB7\x00\x5B\xB5\x00\xCC\x2A\xB4\x00\xA4\x2A\xB4\x00\xA7\xB6\x00\xBA\x2A\xB4\x00" "\xA4\x03\xB6\x00\xB7\xBB\x00\x2B\x59\x04\x05\xB7\x00\x51\x4E\x2A\xB4\x00\xCB\x2D\xB6\x00" "\xBC\x2A\xB4\x00\xCB\x2A\xB4\x00\x66\xB6\x00\xB8\x2A\xB4\x00\xCB\x2A\xB4\x00\x78\xB6\x00" "\xBB\x2A\xB4\x00\xCB\x2A\xB4\x00\xA4\x12\x1A\xB6\x00\x60\x03\x36\x04\xA7\x00\x14\x2A\xB4" $"\x00\x7D\x2A\xB4\x00\x7C\x15\x04\x32\xB6\x00\x61\x84\x04\x01\x15\x04\x2A\xB4\x00\x7A\xA1"$ "\xFF\xE9\x2A\xB4\x00\x7D\x2A\xB6\x00\x62\xBB\x00\x3A\x59\xB7\x00\x50\x3A\x05\xBB\x00\x39" "\x19\x05\x19\x06\xB6\x00\x5F\x57\x19\x05\x2A\xB4\x00\x7D\xB6\x00\x5F\x57\x2A\xB4\x00\xCB" "\x19\x05\x12\x0D\xB6\x00\x60\x2A\x2A\xB4\x00\xAB\x2A\xB4\x00\xAA\x68\xBD\x00\x39\xB5\x00" "\xC9\x03\x36\x07\x03\x36\x08\xA7\x00\x75\x03\x36\x09\xA7\x00\x63\x2A\xB4\x00\xC9\x15\x07" $"\x7F\x2C\x15\x09\x04\x60\x85\x00\x97\x2C\x15\x08\x05\x60\x85\x00\x98\x2C\x04\x85\x00\x95"$ "\x2C\x04\xB5\x00\x96\x2A\xB4\x00\xC9\x15\x07\x32\x04\xB6\x00\xB7\x2B\x2A\xB4\x00\xC9\x15" "\x09\x01\x15\x09\x2A\xB4\x00\xAA\xA1\xFF\x9A\x84\x08\x01\x15\x08\x2A\xB4\x00\xAB\xA1\xFF"

"\x88\x2C\x0F\xB5\x00\xD0\x2C\x0F\xB5\x00\xD1\x2C\x04\xB5\x00\x7F\x2C\x04\xB5\x00\x97\x2C" $\x2B\x2A\xB4\x00\x6D\x2C\xB6\x00\xB9\x2A\x2A\xB4\x00\x6D\xB6\x00\x5F\x57\xBB\x00\x39\x59$ ". "\x12\x19\xB7\x00\x58\x3A\x09\x19\x09\x2A\xB4\x00\xCC\xB6\x00\xBA\x2A\xB4\x00\xB1\x2A\xB4" $"\x00\x99\xB6\x00\xBA\x2A\xB4\x00\x6D\x19\x09\xB6\x00\x5F\x57\x2A\xB4\x00\x6D\xBB\x00\x2C"$ "\x59\x12\x17\x2A\xB4\x00\xBF\x03\xB7\x00\x5C\xB6\x00\x5F\x57\x2A\xB4\x00\x6D\xBB\x00\x2C" $"\x59\x12\x13\x2A\xB4\x00\xBF\x04\xB7\x00\x5C\xB6\x00\x5F\x57\x2A\xB4\x00\x6D\xBB\x00\x2C"$ "\x59\x12\x0F\x2A\xB4\x00\xBF\x03\xB7\x00\x5C\xB6\x00\x5F\x57\x2A\xB4\x00\x6D\x2A\xB4\x00" "\x00\x08\x00\xC3\x00\x0D\x00\xC4\x00\x15\x00\xC6\x00\x24\x00\xC7\x00\x29\x00\xC8\x00\x2E" "\x00\xC9\x00\x33\x00\xCA\x00\x38\x00\xCB\x00\x3D\x00\xCC\x00\x42\x00\xCD\x00\x4A\x00\xCE" "\x00\x53\x00\xCF\x00\x5C\x00\xD1\x00\x64\x00\xD2\x00\x82\x00\xD3\x00\x9C\x00\xD4\x00\xB6" "\x00\xD5\x00\xC1\x00\xD7\x00\xC9\x00\xD8\x00\xD8\x00\xD9\x00\xDB\x00\xDB\x00\xDB\x00\xDB\x "\x00\xE1\x01\x2F\x00\xE2\x01\x38\x00\xE3\x01\x3F\x00\xE4\x01\x48\x00\xE5\x01\x50\x06" "\x01\x5A\x00\xE7\x01\x65\x00\xEA\x01\x75\x00\xEB\x01\x78\x00\xEC\x01\x7E\x00\xEE\x01\x84" "\x00\xFB\x01\xE1\x00\xEE\x01\xED\x00\xEC\x01\xF9\x00\xFE\x01\xFF\x02\x03\x01\x00" $"\x02\x08\x01\x01\x02\x01\x02\x02\x17\x01\x03\x02\x1C\x01\x04\x02\x24\x01\x05\x02\x2D"$ $"\x02\x75\x01\x00\x02\x88\x01\x0e\x02\xA1\x01\x0F\x02\xA0\x00\x00\x02\x01\x0F\x01\x60"$ "\x00\x01\x01\x8B\x00\x00\x02\xB8\x00\x05\x00\x09\x00\x00\x01\xB8\x2A\xB4\x00\x9E\xC6\x01" "\xB3\x2A\x02\xB5\x00\xA8\xBB\x00\x47\x59\x2A\xB6\x00\x85\x2A\xB4\x00\x9E\xB7\x00\x5D\x4C" "\x2B\xB6\x00\xAC\x4D\xBB\x00\x3C\x59\xBB\x00\x3D\x59\x2C\xB6\x00\x89\xB7\x00\x55\xB7\x00" "\x56\x4E\xA7\x01\x73\x2A\xB4\x00\xA8\x9C\x00\x28\x2A\xB4\x00\xA4\xB6\x00\x93\x19\x04\xB6" $"\x00\x77\x9A\x00\x0C\x2A\xB4\x00\xA4\x19\x04\xB6\x00\xBD\x2A\x59\xB4\x00\xA8\x04\x60\xB5"$ "\x00\xA8\xA7\x01\x47\x2A\xB4\x00\xA8\x2A\xB4\x00\xAA\x2A\xB4\x00\xAB\x68\xA2\x01\x37\x19" "\x04\x12\x27\xB6\x00\xC2\x99\x00\x69\x19\x04\xB6\x00\xC6\x3A\x04\x19\x04\x11\xB6" "\x00\xC2\x99\x00\x19\xB2\x00\x94\x3A\x05\xB2\x00\xD2\x3A\x06\x04\x36\x07\x2A\xB4\x00\x99" "\x3A\x08\xA7\x00\x36\x19\x04\x12\x07\xB6\x00\xC2\x99\x00\x19\xB2\x00\x69\x3A\x05\xB2\x00" $"\xD2\x3A\x06\x05\x36\x07\x2A\xB4\x00\xCC\x3A\x08\x19\x04\x04\xB6\x00\xC6\x3A\x04\xA7\x00"$ "\x16\xB2\x00\xD2\x3A\x05\xB2\x00\x69\x3A\x06\x04\x36\x07\x2A\xB4\x00\xA9\x3A\x08\x2A\xB4" "\x00\xC9\x2A\xB4\x00\xA8\x32\xB6\x00\x93\x19\x04\xB6\x00\x77\x9A\x00\x11\x2A\xB4\x00\xC9" "\x2A\xB4\x00\xA8\x32\x19\x04\xB6\x00\xBD\x2A\xB4\x00\xC9\x2A\xB4\x00\xA8\x32\xB6\x00\x83"

"\x2A\xB4\x00\xC9\x2A\xB4\x00\xA8\x32\xB6\x00\x87\x19\x06\xB6\x00\x74\x9A\x00\x11\x2A\xB4"

"\x00\xC9\x2A\xB4\x00\xA8\x32\x19\x06\xB6\x00\xBB\x2A\xB4\x00\xC9\x2A\xB4\x00\xA8\x32\xB6" "\x00\x82\x15\x07\x9F\x00\x11\x2A\xB4\x00\xC9\x2A\xB4\x00\xA8\x32\x15\x07\xB6\x00\xB7\x2A" "\xB4\x00\xC9\x2A\xB4\x00\xA8\x32\xB6\x00\x86\x19\x08\xB6\x00\x76\x9A\x00\x11\x2A\xB4\x00" "\xC9\x2A\xB4\x00\xA8\x32\x19\x08\xB6\x00\xBA\x2A\x59\xB4\x00\xA8\x04\x60\xB5\x00\xA8\x2D" "\xB6\x00\xB2\x59\x3A\x04\xC7\xFE\x89\x2D\xB6\x00\x6B\xB1\x57\xB1\x00\x01\x00\x00\x01\xB5" $\x01\x00\x3F\x00\x01\x01\x00\x00\x00\x00\x00\x3F\x00\x01\x17\x00\x00\x01\x19$ " $"\x01\x20\x00\x2D\x01\x1F\x00\x30\x01\x1E\x00\x34\x01\x22\x00\x37\x01\x24\x00\x3E\x01\x27"$ $"\x00\x40\x01\x28\x00\x56\x01\x29\x00\x60\x01\x24\x00\x63\x01\x28\x00\x73\x01\x30\x00\x70"$ "\x01\x32\x00\x85\x01\x33\x00\x8F\x01\x35\x00\x94\x01\x36\x00\x99\x01\x37\x00\x9C\x01\x38" "\x00\xA2\x01\x33\x00\xA5\x01\x3A\x00\xAF\x01\x3C\x00\xB9\x01\x3E\x00\xBC" $"\x01\x3F\x00\xC2\x01\x3A\x00\xC5\x01\x43\x00\xCA\x01\x44\x00\xCF\x01\x45\x00\xD2\x01\x46\x00\x01\x46\x00\x01\x00\x00\x00\x00\x00\x00\x0$ "\x00\xD8\x01\x48\x00\xE0\x01\x4C\x00\xE8\x01\x4D\x00\xED\x01\x4E\x00\xF0" $"\x01\x4F\x00\xF6\x01\x52\x01\x01\x53\x01\x54\x01\x52\x01\x55\x01\x3A\x01\x56"$ $"\x01\x5C\x01\xA7\x01\x22\x01\xB1\x01\x5F\x01\xB5\x01\x17\x01\xB6\x01\x62\x01\xB7\x01\x15"$ "\x04\x3C\xA7\x00\xCF\xBB\x00\x43\x59\x12\x01\xB7\x00\x59\x4D\x03\x3E\xA7\x00\x27\xBB\x00" $"\x12\x02\xB6\x00\x64\xB6\x00\xCD\x4D\x84\x03\x01\x1D\x2A\xB4\x00\xB1\xB6\x00\x93\xB6\x00"$ $\label{label} $$ \xA6\xA1\xFF\xD1\x2A\x59\xB4\x00\xB0\x04\x60\x5A\xB5\x00\xB0\x2A\xB4\x00\xB1\xB6\x00\x93"$$ "\xB6\x00\xA6\xA1\x00\x08\x2A\x03\xB5\x00\xB0\x2A\xB4\x00\xB1\x2C\xB6\x00\xBD\x2A\xB4\x00" "\xBE\xA7\x00\x31\x10\x0A\x36\x04\x03\x36\x05\xA7\x00\x20\x2A\xB4\x00\x71\x85\xB8\x00\xBE" $"\x2A\xB4\x00\xBF\xB6\x00\x8E\xB6\x00\x8B\x12\x17\xA5\x00\x04\x36\x04\x84\x05\x01\x15"$ "\x05\x15\x04\xA1\xFF\xDF\x2A\xB7\x00\x84\x1B\x99\x00\x09\x03\x3C\x2A\xB6\x00\x73\x2A\xB6" "\x00\xB3\xA7\x00\x09\x57\x04\x3C\xA7\x00\x03\x2A\xB4\x00\xC7\xFF\x30\x2A\x01\xB5\x00" "\xCA\xB1\x00\x01\x00\x05\x00\xC8\x00\xCB\x00\x41\x00\x01\x01\x99\x00\x00\x00\x7E\x00\x1F" "\x00\x00\x01\x69\x00\x02\x01\x6A\x00\x05\x01\x6C\x00\x05\x01\x6E\x00\x0F\x01\x6F\x00\x14" "\x01\x70\x00\x35\x01\x6F\x00\x46\x01\x71\x00\x5E\x01\x72\x00\x63\x01\x73\x00\x6B\x01\x74" $"\\x00\\x7A\\x01\\x75\\x00\\x85\\x01\\x74\\x00\\x88\\x01\\x78\\x00\\x8C\\x01\\x79\\x00\\x92\\x01\\x7B\\x00\\x9A"$ "\x01\x7C\x00\xA9\x01\x7D\x00\xAC\x01\x79\x00\xB6\x01\x80\x00\xBA\x01\x81\x00\xBE\x01\x83" "\x00\xC0\x01\x84\x00\xC4\x01\x86\x00\xC8\x01\x6C\x00\xCB\x01\x87\x00\xCC\x01\x8B\x00\xCE" "\x01\x6C\x00\xD1\x01\x6A\x00\xD8\x01\x8E\x00\xDD\x01\x67\x00\x01\x02\x41\x01\x60\x00\x01" "\x01\x8B\x00\x00\x00\x59\x00\x04\x00\x01\x00\x00\x00\x29\x2A\xB4\x00\xCA\xC7\x00\x19\x2A". "\xBB\x00\x46\x59\x2A\xB7\x00\x57\xB5\x00\xCA\x2A\xB4\x00\xCA\xB6\x00\xC1\xA7\x00\x0A\x2A" "\xB4\x00\xCA\xB6\x00\xB5\x2A\xB7\x00\xC0\xB1\x00\x00\x01\x01\x09\x00\x00\x00\x1E\x00" "\x07\x00\x00\x01\x93\x00\x07\x01\x95\x00\x13\x01\x96\x00\x1A\x01\x93\x00\x1D\x01\x99\x00"

"\x24\x01\x9A\x00\x28\x01\x91\x00\x01\x02\x43\x01\x60\x00\x01\x01\x8B\x00\x00\x00\x00\x00" "\x01\x00\x01\x00\x00\x00\x13\x2A\xB4\x00\xCA\xC6\x00\x0A\x2A\xB4\x00\xCA\xB6\x00\xC8\x2A" $"\x87\x00\xC3\x81\x00\x00\x01\x01\x99\x00\x00\x01\x01\x00\x04\x00\x00\x01\x9F\x00\x07"$ "\x00\x4B\x00\x02\x00\x01\x00\x00\x00\x1F\x2A\xB4\x00\xCA\xC6\x00\x16\x2A\xB4\x00\xCA\xB6" "\x00\xA2\x2A\xB4\x00\xCA\xB6\x00\xC4\x2A\x01\xB5\x00\xCA\x2A\xB7\x00\x72\xB1\x00\x00\x00" "\x01\x99\x00\x00\x00\x1A\x00\x06\x00\x01\xA7\x00\x01\xA9\x00\x01\xAA\x00" "\x15\x01\xAB\x00\x1A\x01\xAD\x00\x1E\x01\xA5\x00\x01\x02\x4C\x01\x71\x00\x01\x01\x8B\x00" $"\x00\x00\x48\x00\x05\x00\x04\x00\x00\x00\x20\x20\x20\x84\x2A\x84\x00\x9C\x84\x00\xD3"$ "\x3D\x2A\xB4\x00\x9C\xB4\x00\x9A\x3E\x2A\xB4\x00\x81\x03\x03\x1C\x1D\xB6\x00\x6A\xB1\x00" "\xB3\x00\x14\x01\xB4\x00\x1F\x01\xAF\x00\x01\x02\x2C\x01\x71\x00\x01\x01\x8B\x00\x00\x00" "\x19\x00\x00\x00\x02\x00\x00\x01\xB1\x00\x00\x01\x01\x99\x00\x00\x00\x01\" "\x00\x00\x00\x4F\x2A\xB6\x00\x91\x4C\x2A\xB4\x00\x9C\xC6\x00\x0F\x2A\xB4\x00\x9C\x2B\xB6" "\x00\x75\x99\x00\x04\xB1\x2A\xBB\x00\x32\x59\x2B\xB7\x00\x54\xB5\x00\x9C\x2A\x2A\x2B\xB4" "\x00\xD3\x2B\xB4\x00\x9A\xB6\x00\x6E\xB5\x00\x9B\x2A\x2A\xB4\x00\x9B\xB6\x00\x88\xB5\x00" "\x81\xB1\x57\xB2\x00\xAD\x12\x0E\xB6\x00\xAF\x2A\x01\xB5\x00\x81\xB1\x00\x01\x00\x18\x00" "\x3F\x00\x40\x00\x3F\x00\x01\x01\x99\x00\x00\x32\x00\x0C\x00\x00\x01\xC3\x00\x05\x01" "\xC4\x00\x17\x01\xC5\x00\x18\x01\xC6\x00\x18\x01\xC8\x00\x24\x01\xC9\x00\x34\x01\xCA\x00" "\x3F\x01\xC6\x00\x40\x01\xCB\x00\x41\x01\xCC\x00\x49\x01\xCD\x00\x4E\x01\xC1\x00\x01\x02" "\x00\x01\x73\x00\x01\x01\x8B\x00\x00\x00\x79\x00\x03\x00\x02\x00\x00\x45\x2B\xB6\x00" "\x92\x2A\xB4\x00\x7D\xA6\x00\x3C\x2A\x2A\xB4\x00\x7D\xB6\x00\x8F\xB5\x00\x5E\x2A\xB4\x00" "\x5E\x9B\x00\x0E\x2A\xB4\x00\x5E\x2A\xB4\x00\x7A\xA1\x00\x0B\x2A\x01\xB5\x00\x9E\xA7\x00" "\x10\x2A\xB4\x00\x7E\x2A\xB4\x00\x5E\x32\xB5\x00\x9E\x2A\xB4\x00\xCA\xB6\x00\xA2\xB1" $"\x00\x00\x01\x01\x01\x09\x00\x00\x00\x02\x00\x00\x00\x00\x00\x01\x04\x00\x00\x01\x06\x00\x16"$ $"\x01\xD7\x00\x28\x01\xD8\x00\x2D\x01\xD7\x00\x30\x01\xD8\x00\x3D\x01\xD8\x00\x44\x01\xD2"$ "\x2A\xB7\x00\x4A\x2A\x12\x24\xB5\x00\xA5\x2A\x12\x1B\xB5\x00\x65\x2A\x12\x23\xB5\x00\x80" "\x2A\x12\x1C\xB5\x00\x67\x2A\x12\x1F\xB5\x00\x79\x2A\x12\x1E\xB5\x00\x70\x2A\x12\x26\xB5" "\x00\xB6\x2A\x12\x1D\xB5\x00\x6C\x2A\x12\x20\xB5\x00\x7B\x2A\x11\x03\xE8\xB5\x00\x71\x2A" $"\times10\times0A\times55\times00\timesAB\times2A\times04\times55\times00\timesAA\times2A\timesBB\times00\times39\times59\times59\times50\times4F\times55\times00\times44\times2A"$ "\xBB\x00\x2D\x59\xB7\x00\x4B\xB5\x00\xBF\x2A\xBB\x00\x3A\x59\xB7\x00\x50\xB5\x00\x6D\x2A" "\xBB\x00\x3A\x59\xB7\x00\x50\xB5\x00\xCB\x2A\xBB\x00\x2E\x59\xB7\x00\x4C\xB5\x00\x7D\x2A" "\xBB\x00\x39\x59\x12\x03\xB7\x00\x58\xB5\x00\xB1\x2A\x11\x03\xE8\xBD\x00\x43\xB5\x00\x68" "\x0A\x00\x21\x00\x10\x00\x22\x00\x16\x00\x23\x00\x1C\x00\x24\x00\x22\x00\x25\x00\x28\x00" "\x26\x00\x2E\x00\x27\x00\x34\x00\x28\x00\x3A\x00\x2D\x00\x47\x00\x2F\x00"

```
"\x4C\x00\xAE\x00\x57\x00\xB9\x00\x62\x00\xBA\x00\x6D\x00\xBB\x00\x78\x00\xBC\x00\x83\x00"
  "\xBD\x00\x90\x01\x12\x00\x9A\x00\x0C\x00\x01\x01\xAD\x00\x00\x00\x02\x01\xBO"
// File user for the web server
class COM_WebPageFileUser : public UTL_FileUser
public:
   COM_WebPageFileUser( int con_sFd, COM_WebServer & server )
        UTL FileUser( bufferSpace, sizeof(bufferSpace) ),
        theServer (server),
        theCon_sFd(con_sFd)
protected:
   virtual void sendFrame(unsigned char *buffer, UR_UINT16 length)
        theServer.sendFrame( buffer, length, theCon_sFd );
                                        // a place to buffer the outgoing data
    unsigned char bufferSpace[1024];
                                //lint !e1725 The server to which to send data
   COM WebServer &theServer;
                                 // socket connection on the server
    int theCon sFd;
};
class WEB_CustomerSupport : public UTL_WebPage
public:
   WEB CustomerSupport(const char*filename)
                                                        UTL WebPage(filename)
    {
        menuFileName = "default.htm";
protected:
   virtual void getBody(UTL_FileUser & dest,int optionCount, const char *options[],const
char *filename)
    {
        (void)optionCount;
        (void) options;
        (void) filename;
        UTL_WebPage::Table t(2,dest);
        t.startBannerCell();
        dest.puts("GE Power Management");
        t.startHeadingCell("right");
        dest.puts("Address: ");
        t.startCell("left");
        dest.puts("215 Anderson Ave. <BR>");
        dest.puts("Markham, Ontario<BR>");
        dest.puts("Canada L6E 1B3<BR>");
        t.startHeadingCell("right");
        dest.puts("Phone: ");
        t.startCell("left");
        dest.puts("(905) 294-6222");
        t.startHeadingCell("right");
        dest.puts("Fax: ");
        t.startCell("left");
        dest.puts("(905) 294-2098");
        t.startHeadingCell("right");
        dest.puts("Email: ");
        t.startCell("left");
        dest.puts("<A HREF=mailto:info.pm@indsys.ge.com>info.pm@indsys.ge.com</A>");
        t.startHeadingCell("right");
        dest.puts("Internet: ");
        t.startCell("left");
        dest.puts("<A
HREF=http://www.GEindustrial.com/pm>http://www.GEindustrial.com/pm</A>");
    virtual void printTitle(UTL FileUser & dest,int optionCount, const char
*options[],const char *filename)
    {
        (void) optionCount;
        (void) options;
        (void) filename;
        dest.puts( "Customer Support Information" );
    }
```

```
};
#include "COM ModbusAddress.h"
class DB_MemoryMapWebPage : public UTL_WebPage
public:
    DB MemoryMapWebPage(const char*filename)
       : UTL_WebPage(filename)
        menuFileName = "default.htm";
protected:
   virtual void getBody(UTL_FileUser & dest,int optionCount, const char *options[],const
char *filename)
    {
        (void) filename;
        UR MODULE lastModule=UR_MODULE(-1), selectedModule=UR_MODULE(-1);
        UR_BOOLEAN summary = UR_FALSE;
        if( optionCount )
            int m = 0;
            (void) sscanf(options[0], "%d", &m);
            selectedModule = (UR_MODULE)m;
        else
            summary = UR_TRUE; // if no module specified, present a summary
        UTL_WebPage::Table t(3,dest);
        if( summary )
            t.startBannerCell();
dest.puts( "MEMORY MAP SUMMARY");
            t.startHeadingCell();
            dest.puts("Address");
            t.startHeadingCell("left");
            dest.puts("Module");
            t.startHeadingCell();
            dest.puts("Array Size");
        else
            t.startTable(7);
            t.startBannerCell();
            dest.printf( "MEMORY MAP FOR \"%.80s\"",
                                 SYS Product::find()->getName(selectedModule) );
            t.startHeadingCell();
            dest.puts("Address");
            t.startHeadingCell("left");
            dest.puts("Name");
            t.startHeadingCell();
            dest.puts("Type");
            t.startHeadingCell();
            dest.puts("Min");
            t.startHeadingCell();
            dest.puts("Max");
            t.startHeadingCell();
            dest.puts("Value");
            t.startHeadingCell();
            dest.puts("Unit");
        DB_DataItem * d = 0;
        UR UINT16 moduleIndex = 0;
        UR_UINT16 arrayIndex = 0;
        UR_UINT16 nextAddress = 0xFFFF; // address of next array element
        UR_UINT16 addr=0;
        do
            COM ModbusAddress * ma = COM_ModbusAddress::find(addr);
            if( ma )
            {
                                              // possibly next array index for current item
                     addr==nextAddress
                     || ma->getItem() != d
                                              // new data item
```

```
|| ma->getModuleIndex() != moduleIndex // next module index for
current item
                {
                    if( ma->getItem() != d )
                        arrayIndex = 0;
                        d = ma->getItem();
                        moduleIndex = ma->getModuleIndex();
                    else if( ma->getModuleIndex() != moduleIndex )
                        arrayIndex = 0; // reset array index at start of new module
                        moduleIndex = ma->getModuleIndex();
                    else
                        arrayIndex++; // traversing item array in same module of same
item
                    if( summary )
                        if( d->module != lastModule )
                             t.startCell();
                            dest.printf("%04X",addr);
                            t.startCell("left");
                            dest.printf("<A HREF=%s?%d>%s</A>",
                                 getFileName(),
                                 (int)d->module,
                                 SYS_Product::find()->getName(d->module)
                             t.startCell();
                            dest.printf("%d",(int)SYS_Product::find()->getSize(d-
>module));
                            lastModule = d->module;
                    else if( d->module == selectedModule )
                        nextAddress = addr + (d->getSize()+1)/2;
                                                                   // here's where the
next array element is
                        char s[100];
                        char a[100];
                        UR UINT16 c;
                        t.startCell();
                        dest.printf("%04X",addr);
                        t.startCell("left");
                        d->getFormattedName(UR_TRUE,&c,&s[0],moduleIndex,arrayIndex);
                        dest.puts(&s[0]);
                        t.startCell();
                        if( d->attrib.eeprom )
                            dest.puts("Read/Write Setting");
                        else if(d->attrib.write)
                            dest.puts("Writable Actual");
                        else if( d->attrib.sram )
                            dest.puts("Non-volatile Actual");
                        else
                            dest.puts("Read Only");
                        t.startCell();
                         (void)d->getMinimum(&s[0]);
                         (void)d->toAscii(&c,a,s,moduleIndex,arrayIndex,0);
                        if( !*a )
                             t.setFontBold();
                             dest.puts("(?)");
                         else
                             dest.puts(a);
                         t.startCell();
                         (void)d->getMaximum(&s[0]);
                         (void)d->toAscii(&c,a,s,moduleIndex,arrayIndex,0);
                         if( !*a )
                         {
                             t.setFontBold();
                             dest.puts("(?)");
```

```
else
                              dest.puts(a);
                          t.startCell();
                          (void) d->get(s,moduleIndex,arrayIndex,0);
                          (void)d->toAscii(&c,a,s,moduleIndex,arrayIndex,0);
                          if( !*a )
                          {
                              t.setFontBold();
                              dest.puts("(?)");
                          else
                              dest.puts(a);
                          t.startCell();
                          if( ! * webString(a,d->getUnit(moduleIndex,arrayIndex,0)) )
                              strcpy(a, " ");
                          dest.puts(a);
                     }
                 }
        } while( ++addr );
    virtual void printTitle(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename)
    {
         (void) optionCount;
         (void)options;
         (void) filename;
        dest.puts( "Modbus Memory Map");
    }
};
#include "memLib.h"
#include "DSP_Card.h"
#include "UTL_TaskDataPointer.h" // focass WEB_MiscStats : public UTL_WebPage
                                      // for testing only
public:
                                                     UTL WebPage (filename), tdp(30)
    WEB MiscStats(const char*filename)
        menuFileName = "DiagnosticsMenu.htm";
protected:
    virtual void getBody(UTL_FileUser & dest,int optionCount, const char *options[],const
char *filename)
         (void)optionCount;
         (void) options;
         (void) filename;
        UTL WebPage::Table t(2,dest);
         for( int i=0; i<N_DSPS; i++ )</pre>
             DSP Card * d = DSP_Card::find(i);
             if( d )
             {
                 t.startCell("right");
                 t.setFontBold();
                 dest.printf("DSP %d usage:",i);
                 const DSP_State & p = d->getDspState();
                 t.startCell();
                 dest.printf("%.1f%%",float(p.Dsp Usage)/10.0);
         t.startCell("right");
         t.setFontBold();
         dest.printf("Largest Free Memory Block");
         t.startCell();
dest.printf( "%d bytes",memFindMax());
         static int myNumber=1;
         char * myName = (char*)(tdp.get());
         if( !*myName )
             (void) sprintf(&myName[0], "TASK %d", myNumber++);
         t.startCell("right");
```

```
t.setFontBold();
        dest.puts("HTTP Connection Number:");
        t.startCell();
        dest.puts(myName);
    virtual void printTitle(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename)
         (void)optionCount;
         (void) options;
         (void) filename;
        dest.puts( "Miscellaneous Diagnostics" );
    UTL TaskDataPointer tdp;
                                   // for testing only
};
#define SOCKET_ERROR ERROR
                              // the number 2
// the number 3
#define INTEGER_2
#define INTEGER_3
                     3
                              // the number 4
#define INTEGER 4
// The following definitions allow incorporation of socket calls into both the
// GNU and WIN32 builds.
#ifdef WIN32
    // WIN32 version of socket calls.
                      SOCKET_CALL_INET_NTOA_B(unsigned long inetAddress, char *pString);
    extern void
                      SOCKET CALL INET NTOA B(in addr inetAddress, char *pString)
    static void
        SOCKET_CALL_INET_NTOA_B(inetAddress.s_addr, pString);
                     SOCKET_CALL_SETSOCKOPT (int s, int level, int optname, char *optval,
    extern STATUS
int optlen);
                      SOCKET CALL SEND (int s, char *buf, int bufLen, int flags);
    extern int
                     SOCKET_CALL_ACCEPT (int s, struct sockaddr *addr, int *addrlen);
SOCKET_CALL_LISTEN (int s, int backlog);
SOCKET_CALL_BIND (int s, struct sockaddr *name, int namelen);
    extern int
    extern STATUS
    extern STATUS
                     SOCKET_CALL_SOCKET (int domain, int type, int protocol);
SOCKET_CALL_RECV (int s, char *buf, int bufLen, int flags);
    extern int
    extern int
                      SOCKET_CALL_CLOSE (int fd);
    extern STATUS
                     SOCKET_CALL_SHUTDOWN(int s, int how);
SOCKET_CALL_CONNECT(int s, struct sockaddr * name, int namelen);
    extern STATUS
    extern STATUS
    // GNU version of socket calls -- simply map them to the VxWorks function names.
#define SOCKET_CALL_INET_NTOA_B inet_ntoa_b
#define SOCKET_CALL_SETSOCKOPT setsockopt
#define SOCKET_CALL_SEND send
#define SOCKET_CALL_ACCEPT accept
#define SOCKET_CALL_LISTEN listen
#define SOCKET_CALL_BIND bind
#define SOCKET_CALL_SOCKET socket
#define SOCKET CALL RECV recv
#define SOCKET CALL CLOSE close
#define SOCKET CALL SHUTDOWN shutdown
#define SOCKET_CALL_CONNECT connect
#endif
COM_WebServer * COM_WebServer::the_COM_WebServer = 0;
// FUNCTION
                   COM WebServer::COM WebServer
// DESCRIPTION TopPort class constructor.
COM WebServer::COM WebServer(void)
    the COM WebServer = this;
```

```
isInitialized = UR_FALSE;
    pleaseKillMe = false;
    connectionCount = 0;
    numRunningTasks = 0;
    for( int i=0; i<MAX_HTTP_CONNECTIONS; i++ )</pre>
        connectionTimers[i] = new UTL_1msTimer(SOCKET_TIMEOUT * 1000); // convert to
milliseconds
    IP_Address.registerForNotification(this);
    // Create our web pages
    (void)new UTL_StaticFile("bug.gif", (unsigned char*)&GifBug, sizeof(GifBug));
(void)new UTL_StaticFile("UR_Grid.class", (unsigned char*)&UR_GridClass,
sizeof(UR GridClass));
    (void)new DB MemoryMapWebPage("memoryMap.htm");
    (void) new UTL WebMenu("DeviceInfoMenu.htm", "default.htm", "Device Information Menu"); (void) new UTL WebMenu("DiagnosticsMenu.htm", "default.htm", "Diagnostics
Menu", FACTORY_LEVEL);
    (void) new WEB_MiscStats("MiscStats.htm");
    (void)new WEB CustomerSupport("CustomerSupport.htm");
}
// FUNCTION
                COM WebServer::~COM WebServer
// DESCRIPTION COM_WebServer class destructor.
COM WebServer::~COM_WebServer()
    int i;
                                      // let tasks know we're all done
    pleaseKillMe = true;
    // Kill all the connected sockets
    for( i=0; i<MAX_HTTP_CONNECTIONS; i++ )</pre>
        connectionTimers[i]->stop();
        connectionTimers[i]->setTimeDelay(1);
                                                  // this should cause existing connections
        connectionTimers[i]->start();
to die
    // Kill the unconnected sockets
    for( i=0; i<MAX_HTTP_CONNECTIONS; i++ )</pre>
        // Connect to each socket, then disconnect -- the receive task will see the
        // pleaseKillMe flag, and quit.
        // socket structure to use vxWorks TCP-functions without causing a pclint warning
        union socket_stuff
             sockaddr_in in_Addr;
            sockaddr
                       sock_Addr;
        };
        union socket_stuff server_stuff;
                                              // server socket address
                                              // size of socket address structure
// "localhost" address (127.0.0.1)
                              sockAddrSize;
        u long localhost = 0x7f000001;
        int clientFd;
                                               // client socket
        // Create client socket
        clientFd = SOCKET_CALL_SOCKET (AF_INET, SOCK STREAM, 0);
        if (clientFd == SOCKET_ERROR)
        {
            printf("\nSocket creation error.\n");
             continue;
        // set up the local address
        sockAddrSize = sizeof (server stuff.in_Addr);
        bzero ( (char *)&server stuff.in Addr, sockAddrSize );
server_stuff.in Addr.sin_family = AF_INET;
        server_stuff.in_Addr.sin_port = htons (SERVER_PORT_NUM);
```

```
server stuff.in Addr.sin_addr.s_addr = htonl (localhost);
     // "Open and shut case"
     // Once we've connected, the Rx task will die, so we can close our end
     // of the socket. If we didn't connect, we close anyway.
     (void) SOCKET CALL CONNECT(clientFd, &server_stuff.sock_Addr, sockAddrSize);
     SOCKET CALL_CLOSE(clientFd);
  }
  while ( numRunningTasks )
     taskDelay(1);
               // give the tasks a chance to die
  // Now clean up a bit
  for( i=0; i<MAX_HTTP_CONNECTIONS; i++ )
     delete connectionTimers[i];
     connectionTimers[i] = 0;
  }
  the_COM_WebServer = 0;
// FUNCTION
            COM WebServer::sendFrame
// DESCRIPTION
            Initiates transmission of a frame.
void COM_WebServer::sendFrame
  unsigned char *buffer, // pointer to response buffer
                     // number of bytes in response buffer
  UR_UINT16 length,
  int con sFd
                  // TCP connection number
)
{
  if (SOCKET_CALL_SEND(con_sFd, (char *) buffer, length, 0) == SOCKET_ERROR)
    printf("\nSocket send error.\n");
             // don't hold up higher-priority activities
  taskDelay(2);
}
// This function calls the connect task in a portable way.
{
  obj->numRunningTasks++;
  obj->connect Task();
  obj->numRunningTasks--;
  return 0;
int COM WebServer::call_read_Task(
  COM_WebServer *
                             // object in which to call the task function
                 obj,
                connectionNumber
                             // connection number
  int
{
  obj->numRunningTasks++;
  obj->read_Task( connectionNumber);
  obj->numRunningTasks--;
  return 0;
}
// FUNCTION
            COM WebServer::connect Task
// DESCRIPTION
            Listen for connections and spawn read tasks when connections
            are established.
```

```
void COM WebServer::connect_Task()
    // socket structure to use vxWorks TCP-functions without causing a pclint warning
    union socket_stuff
    {
        sockaddr_in in_Addr;
        sockaddr_
                   sock_Addr;
    };
    union socket_stuff server stuff;
                                          // server socket address
                                         // size of socket address structure
                         sockAddrSize;
    int
                         ix = 0;
                                          // counter for read task names
                                         // name of read tasks
                         task name[16];
    char
                         optval;
                                          // socket options
    int
    // set up the local address
    sockAddrSize = sizeof (server_stuff.in_Addr);
    bzero ( (char *)&server_stuff.in_Addr, sockAddrSize );
    server_stuff.in_Addr.sin_family = AF_INET;
server_stuff.in_Addr.sin_port = htons (SERVER_PORT_NUM);
    server_stuff.in_Addr.sin_addr.s_addr = htonl (INADDR_ANY);
    // create a TCP-based socket
    if ((sFd = SOCKET_CALL_SOCKET (AF_INET, SOCK_STREAM, 0)) == SOCKET_ERROR)
    {
        printf("\nSocket creation error.\n");
        return;
    }
    // set socket options
  optval = 1; // SO_KEEPALIVE on
      SOCKET_CALL_SETSOCKOPT (sFd, SOL_SOCKET, SO_KEEPALIVE, (caddr_t) &optval, sizeof
(optval));
    optval = 1; // TCP_NODELAY on
SOCKET_CALL_SETSOCKOPT (sFd, SOL_SOCKET, TCP_NODELAY, (caddr_t) &optval, sizeof
(optval));
    optval = 1; // SO REUSEADDR on
    SOCKET_CALL_SETSOCKOPT (sFd, SOL_SOCKET, SO_REUSEADDR, (caddr_t) &optval, sizeof
(optval));
    struct linger lng;
    lng.l_linger = 0;
lng.l_onoff = 1;
                              // zero timeout on linger
    SOCKET_CALL_SETSOCKOPT (sFd, SOL_SOCKET, SO_LINGER, (caddr_t) &lng, sizeof (lng));
    // bind socket to local address
    if (SOCKET_CALL_BIND (sFd, &server_stuff.sock_Addr, sockAddrSize) == SOCKET_ERROR)
        printf("\nSocket bind error.\n");
        SOCKET CALL CLOSE (sFd);
        return;
    }
    // create queue for client connection requests
    if (SOCKET_CALL_LISTEN (sFd, MAX_HTTP_CONNECTIONS) == SOCKET_ERROR)
        printf("\nSocket listen error.\n");
        SOCKET_CALL_CLOSE (sFd);
        return;
    }
    for( ix=0; ix<MAX_HTTP_CONNECTIONS; ix++ )
        sprintf (task_name, "WebRx%d", ix);
         (void) taskSpawn(task_name, SYS_Application::utilityPriority, 0, 10000,
             (FUNCPTR) call_read_Task, (int) this, ix, 0, 0, 0, 0, 0, 0, 0);
    // Now loop forever checking the timers for all the connections
    int t = sysClkRateGet(); // once per second should do it
    while ( sFd != SOCKET_ERROR ) // keep going until main socket is closed by
destructor
    {
        taskDelay(t);
```

```
for( int i=0; i<MAX_HTTP_CONNECTIONS; i++ )</pre>
           if( connected_sFd[i] != SOCKET_ERROR && connectionTimers[i]->isElapsed() )
               // timer elapsed - kill the connection, but not in a polite way
   #if DEBUG HTTP
               printf("http %d: timed out -- shutting down\n", i);
   #endif
               (void) SOCKET_CALL_SHUTDOWN(connected_sFd[i],2);
       // If we're shutting down and this is the last task, die
       if( pleaseKillMe && numRunningTasks<=1 )</pre>
           break;
   printf("Web server connect task is finished\n");
   SOCKET CALL CLOSE (sFd);
}
// Create a "page not found" page when the browser has requested a file
// which either doesn't exist or is inaccessible.
void COM_WebServer::notFoundPage(int connected_sFd)
   char response[1000];
   sprintf( response,
       "<HTML>\n"
       "<HEAD>"
       "<meta http-equiv=\"refresh\" content=\"5\">"
       "<TITLE>Page Not Found</TITLE></HEAD>\n"
       "<BODY BGCOLOR=\"#FFFFFF\">\n"
       "<H1>PAGE NOT FOUND</H1><BR><BR>\n"
       "<HR><STRONG><A HREF=default.htm>Click Here For The Main Menu</A></STRONG><HR>\n"
       "</BODY></HTML>\n\"
       );
#ifdef WIN32
   printf("Sending----\n%s\n----", response);
#endif
   sendFrame( (unsigned char *)response,strlen(response),connected_sFd );
}
// FUNCTION
                COM WebServer::read Task
                Wait for data from a socket and then send it to the attached
// DESCRIPTION
                protocol application.
//
11
void COM WebServer::read_Task
(
                      connectionNumber // connection number -- identifies this task
   int
)
   UTL_TaskDataBlock taskDataBlock;  // each task gets a data block for task-specific
                                     // socket addr buffer for inet_ntoa_b()
   char inet name[18];
   // socket structure to use vxWorks TCP-functions without causing a pclint warning
   union socket_stuff
       sockaddr_in in_Addr;
                 sock Addr;
       sockaddr_
   1:
   union socket_stuff client_stuff;
                                     // client socket address
                      sockAddrSize = sizeof(client_stuff);
                                                          // size of socket address
   int
structure
   UTL WatchDog wd(10000, false);
   // accept new connect requests and spawn tasks to process them
   while ( sFd != SOCKET ERROR)
       if ((connected sFd[connectionNumber] = SOCKET CALL_ACCEPT (sFd,
&client_stuff.sock_Addr, &sockAddrSize)) == SOCKET_ERROR)
           printf("\nSocket #%d accept error.\n", connectionNumber);
           SOCKET CALL CLOSE (sFd);
```

```
break;
        // Shut down if asked to do so
        else if( pleaseKillMe )
            SOCKET CALL CLOSE(connected.sFd[connectionNumber]);
        wd.kick();
        struct linger lng;
        lng.l_linger = 0;
                                // zero timeout on linger
        lng.l_onoff = 1;
SOCKET_CALL_SETSOCKOPT (connected_sFd[connectionNumber], SOL_SOCKET, SO_LINGER,
(caddr_t) &lng, sizeof (lng));
        // Start the dead connection timer
        connectionTimers(connectionNumber)->start();
        // convert the client address to internet address form
        SOCKET_CALL_INET_NTOA_B( client_stuff.in_Addr.sin_addr, inet_name );
        connectionCount++;
#if DEBUG_HTTP
        printf ("\nSocket #%d open. Connection count = %d\n", connectionNumber,
connectionCount);
#endif
                            clientRequest[1200];// request/message from client
        unsigned char
                                                 // number of bytes read
                            nRead;
        // read client request and process messages.
        while( (nRead = SOCKET_CALL_RECV( connected_sFd[connectionNumber],
                                                 (char*)clientRequest,
                                                 sizeof(clientRequest)-1,
                                                 0)) > 0)
        {
            wd.kick();
            // Shut down if asked to do so
            if( pleaseKillMe )
                break;
            // Got something from the client -- process it.
                                        // null terminate
            clientRequest[nRead] = 0;
            // re-start the dead connection timer
            connectionTimers[connectionNumber]->start();
#if DEBUG HTTP>1
            // message display (enable for debugging if required)
            printf ("\nMESSAGE FROM CLIENT on #%d (Internet Address %s, port %d, length
%d, sFd %d) \n",
                {\tt connectionNumber, address, port, nRead, connected\_sFd[connectionNumber]);}
            for (int i=0; i<nRead; i++)
                printf("%u ", clientRequest[i]);
            printf("\n");
#endif
#ifdef WIN32
            printf("HTTPD #%d GOT ::::::::::\n%s\n----\n",
                connectionNumber, (char*)clientRequest );
#endif
            char fileNameToGet[500] = "/";
            if( ! strncmp((char*)clientRequest, "GET", 3) )
                // check the client's authorization
                clientPassword[0] = 0; // kill off the existing password information
                char *p = (char*)clientRequest;
                char *line = p;
                char *tokens[20];
                int tokenNumber = 0;
                tokens[0] = p;
                do
                    switch( *p )
                    case '\r':
```

```
*p = 0;
                         break;
                     case ' ':
                     case '\t':
                     case ':':
                         // end of a token
                         *p = 0;
                         if( tokens[tokenNumber] == p )
                             tokens[tokenNumber]++; // move token past the whitespace
                         else
                             tokens[++tokenNumber] = p+1;
                                                                        // go to the next
token
                         break;
                     case '\n':
                     case '\0':
                         // end of a line
                         *p = 0;
                         // implies end of a token
                         if( tokenNumber >= 2 && ! strcmp(tokens[0], "GET") )
                         {
                             strncpy(fileNameToGet, tokens[1], sizeof(fileNameToGet)-1);
                             fileNameToGet[sizeof(fileNameToGet)-1] = 0;
                         else if( tokenNumber >= 3 )
                             // three tokens means it might be an authorization line
if( (! strcmp(tokens[0],"Authorization")
                                  && (! strcmp(tokens[1], "Basic") ) ) )
                                  // Great! Got a basic authorization line
                                  // Third token is base-64 encoded password
                                  int pwSize = b64_decode(tokens[2], (unsigned char
*)clientPassword, sizeof(clientPassword));
                                  clientPassword[pwSize] = 0;
                             }
                         line = p+1;
                         tokenNumber = 0;
                         tokens[0] = line;
                         break;
                     default:
                         break;
                     }
                     p++;
                } while(*p);
                char * optionString = (char*)fileNameToGet;
                while( *optionString )
                     if( *optionString == '?' )
                         *optionString++' = 0; // found bounadary between file name and
options
                         break;
                     optionString++;
                 COM_WebPageFileUser u( connected_sFd[connectionNumber], *this );
                if(! strcmp(fileNameToGet,"/"))
    strcpy(fileNameToGet,"/default.htm");
UTL_FileSource * f = UTL_FileSource::find(fileNameToGet);
                 if( f && f->isAccessible() )
                     const char *optionArray[20];
                     int optionCount = 0;
                     char *p = optionString;
                     if(*p)
                         optionArray[optionCount++] = p;
                     while( *p )
                         if( *p++ == '&' ) // look for option separator
                              optionArray[optionCount++] = p;
                     }
```

```
f->get(u,optionCount,optionArray,fileNameToGet);
                  u.flush();
               else
                  notFoundPage(connected sFd[connectionNumber]);
    #if DEBUG HTTP
               printf("http %d: transmission complete\n", connectionNumber);
   #endif
               break;
           }
       if (nRead == SOCKET ERROR)
                                               // error from read()
           printf ("\nSocket #%d read error.\n", connectionNumber);
       // Stop the dead connection timer, since we have come out cleanly
       connectionTimers[connectionNumber] -> stop();
       // interlock to avoid fight with timer
       int savedFd = connected_sFd[connectionNumber];
       STATUS err;
       if( savedFd != SOCKET ERROR )
   #if DEBUG HTTP
              printf("http %d: normal shutdown", connectionNumber);
   #endif
           err = SOCKET_CALL_SHUTDOWN(savedFd,1); // shut down send side
           if( err == SOCKET_ERROR )
              printf("\nhttp %d: shutdown error\n", connectionNumber);
           // make sure all the reads are finished
           // (seems to be necessary to clean out the final ACK)
           while( SOCKET_CALL_RECV( savedFd, (char*)clientRequest,
                                   sizeof(clientRequest)-1, 0) > 0)
           {
               // just loop
               wd.kick(); // kick again to re-start the watchdog
   #if DEBUG HTTP
              printf(".");
   #endif
           (void)SOCKET_CALL_SHUTDOWN(savedFd,2); // shut down everything
   #if DEBUG HTTP
              printf("close...");
   #endif
           connected_sFd[connectionNumber] = SOCKET_ERROR; // mark the socket closed so
the timer can't mess us up
           err = SOCKET_CALL_CLOSE (savedFd);
                                                  // close server socket connection
           if( err == SOCKET_ERROR )
               printf("\nhttp %d: close error\n", connectionNumber);
#if DEBUG HTTP
       printf ("\nSocket #%d closed.\n", connectionNumber);
#endif
       connectionCount--;
}
// First time in, starts the web connect task
void COM_WebServer::acceptNotification
    DB_NotificationSource *source, // not used
   int param
    (void) source; // use this parameter to avoid a compiler warning
    (void) param; // use this parameter to avoid a compiler warning
    if (isInitialized == UR_FALSE)
```

```
sprintf(tName, "WebConnectTask");
           taskSpawn(tName, 40, 0, 4000, (FUNCPTR) call_connect_Task, (int) this, 0, 0, 0, 0,
0, 0, 0, 0, 0);
           isInitialized = UR_TRUE;
      }
}
// Base-64 decoding. This represents binary data as printable ASCII
// characters. Three 8-bit binary bytes are turned into four 6-bit
// values, like so:
//
       [1111111] [22222222] [33333333]
11
11
       [111111] [112222] [222233] [333333]
//
// Then the 6-bit values are represented using the characters "A-2a-z0-9+/".
const int COM_WebServer::b64_decode_table[256] =
                                                                             /* 00-0F */
      /* 10-1F */
      /* 20-2F */
     52,53,54,55,56,57,58,59,60,61,-1,-1,-1,-1,-1,-1,
                                                                             /* 30-3F */
     -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, -1, -1, -1, -1, -1,
                                                                             /* 40-4F */
                                                                             /* 50-5F */
                                                                             /* 60-6F */
      -1,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
      41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, -1; -1, -1, -1, -1,
                                                                             /* 70-7F */
     /* 80-8F */
                                                                             /* 90-9F */
      /* A0-AF */
      -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
      /* BO-BF */
                                                                             /* CO-CF */
     /* DO-DF */
     /* E0-EF */
      /* FO-FF */
};
// Do base-64 decoding on a string. Ignore any non-base64 bytes.
// The decoded size will
// be at most 3/4 the size of the encoded, and may be smaller if there
// are padding characters (blanks, newlines).
// RETURNS: the actual number of bytes generated.
int COM_WebServer::b64_decode(
                                           const char* str,
                                                                         // source string
                                           unsigned char* space, // destination buffer
                                                                          // size of destination buffer
                                           int size
                                           )
{
     const char* cp;
     int space_idx, phase;
     int d, prev_d;
     unsigned char c;
     space idx = 0;
     phase = 0;
     prev d = 0;
      for ( cp = str; *cp != '\0'; ++cp )
           d = b64 decode table[*cp];
           if (d!=-1)
                 switch ( phase )
                 default:
                 case 0:
                       ++phase;
                      break;
                 case 1:
                      c = ( (prev d << 2) | ( (d & 0x30) >> 4) );
                       if ( space_idx < size )
                            space[\overline{space_idx++}] = c;
                       ++phase;
                      break;
                 case 2:
                       c = ( ( prev_d & 0xf ) << 4 ) | ( ( d & 0x3c ) >> 2 ) );
                       if ( space_idx < size )
```

Listing 3: UTL_FileSource.h

```
*****************
 * Copyright (C) General Electric Co. GE Confidential and Proprietary
 * DESCRIPTION Generic file source class
#ifndef _UTL_FILESOURCE_H_
#define _UTL_FILESOURCE_H_
#include "SYS_Types.h"
#include "UTL_FileUser.h"
#include "DB_DataItem.h"
                               // for DB SECURITY LEVEL
// File source class -- provides data for a file, which may then be read
// by a UTL_FileUser object. The UTL_FileUser objects are associated with
// all the channels through which we might want to read files: modbus, UCA,
// web server, etc. Subclasses define specific types of files.
// <BR> Here are some key points about this class:
// <UL>
// <LI>
         File contents are not stored anywhere -- they are created as needed
         File contents are dynamic (i.e., can be different each time you read it)
Reading is a "Push" operation, using, for example, UTL_FileUser::printf()
// <LI>
         to dump the entire contents of a file to a UTL_FileUser object when
11
          requested to do so.
//
         Files are read by calling the "get" function % \left( 1\right) =\left( 1\right) ^{2}
// <LI>
         Files can be located by filename using the "find" function
// <LI>
         Subclasses can override the "isOne" function to have variable filenames.
// <LI>
         For example, oscillography file names can contain an embedded trace number (eg: OSC1234.CFG). The default function checks for an exact,
11
11
//
          case-insensitive match.
         You can simulate directories by embedding slashes in the filenames
         The "isAccessible" function tells whether the file is accessible under
// <LI>
          whatever security arrangements are appropriate for the specific file.
11
          For example, some files may only be accessible when factory service
11
          is enabled.
         The "printTitle" function may be overridden to provide a title for
          the file. The title occurs in the DIR.TXT file, and as a title for web pages. The base class version prints the filename.
//
11
         Many of the functions take optionCount and options as arguments.
   <LI>
          The options are gathered by the specific protocol which is reading
11
          the file, looking for whatever delimiters are appropriate for
          each protocol. These options may be used to select the specific data or format in which the information will be provided. For example,
//
          reading a "memory map" object with no options could read a summary memory map, while specifying options to the same object could provide, say,
//
//
         the memory map for a specific module.
Files can be "pulled" (read some data, then some more, etc.) rather
than "pushed" (initiate transmission of the complete file) using
11
// <LI>
11
          the UTL_FilePuller file user class, although task and memory overhead
//
          is incurred when you do so.
           class UTL_FileSource
public:
```

```
UTL_FileSource( const char *filename, DB_SECURITY_LEVEL anAccessLevel=NO_LEVEL);
       virtual ~UTL_FileSource();
       // Overridable function gets the file into a file user, by calling the
       // write and puts functions in the UTL_FileUser.
       virtual void get(
                                    // put output here
           UTL FileUser & dest,
            int optionCount, // number of options
        const char *options[],
                                // options, if any
                                // filename being got, in case the filename contains
        const char *filename
options
    virtual UR BOOLEAN isOne( const char * filename );
    // Get the file name
    const char * getFileName(void) { return theFileName; }
    // Get a pointer to the first file
    static UTL_FileSource * getFirst(void) { return head; }
    // Get a pointer to the next file
    UTL_FileSource * getNext(void) { return next; }
    UR BOOLEAN isAccessible(void);
       // Find the object corresponding to the given filename
       static UTL_FileSource * find(const char *filename);
    static void deleteAll(void);
       virtual void printTitle(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename);
    // access function for menu file name
    const char * getMenuFileName(void) { return menuFileName; }
protected:
   UR_BOOLEAN filenameCharsMatch(char c1, char c2); // compare two filename characters .
UR_BOOLEAN filenameMatch( const char * basename, const char * filename, unsigned &
number );
       char * theFileName;
       UTL FileSource * next;
                                              // pointer to next instance
    static UTL_FileSource * head; // pointer to first instance
const char * menuFileName; // include this file in the named menu file
    DB SECURITY_LEVEL accessLevel; // access level required to read this file
};
#endif
Listing 4: UTL FileSource.cpp
                                         *********
 * Copyright (C) General Electric Co. GE Confidential and Proprietary
 * DESCRIPTION File source class
#include "UTL_FileSource.h"
#include "UTL_FileUser.h"
#include "MMI_Application.h"
#include <assert.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
UTL FileSource * UTL FileSource::head = 0;
                                                  // becomes UR TRUE when directory is
static UR_BOOLEAN directoryCreated = UR_FALSE;
class UTL_FileSourceDir : public UTL_FileSource
public:
    UTL FileSourceDir( const char *filename )
        : UTL FileSource(filename)
    void printTitle(
```

```
UTL FileUser & dest,
                                 // destination object
        int optionCount,
        const char *options[],
        const char *filename
    {
        (void) optionCount;
        (void) options;
        (void) filename;
        dest.puts("A directory of all the files in the relay");
    }
           UTL_FileUser & dest,
                                   // put output here
                              // number of options
           int optionCount,
        const char *options[],
                                // options, if any
        const char *filename
                                 // filename being got, in case the filename contains
options
    {
        (void) optionCount;
        (void) options;
        (void) filename;
        dest.puts("File directory\r\n\r\n");
        UTL FileSource * it = UTL_FileSource::getFirst();
        while( it )
            if( it->isAccessible() )
                const char *dummyOptions[1];
dest.printf("%-50.200s: ", it->getFileName() );
                it->printTitle(dest, 0, dummyOptions, it->getFileName());
                dest.puts("\r\n");
            it = it->getNext();
   }
};
// Constructor -- record the file info.
// If your filename has a leading slash or backslash, it gets removed, so that
// all files are relative to the root ("/") directory.
UTL_FileSource::UTL_FileSource(
                        const char *filename,
                                                             // file name
                                                             // access level
                        DB_SECURITY_LEVEL anAccessLevel
       // insert into linked list
       next = head;
       head = this;
    assert( filename );
    if( *filename == '\\' || *filename == '/' )
        filename++;
                       // skip over leading slash or backslash, if supplied
       theFileName = new char[strlen(filename)+1];
    assert(theFileName);
       strcpy(theFileName, filename);
    menuFileName = "";
                                      // subclasses may override as required
    accessLevel = anAccessLevel;
    if( !directoryCreated )
        directoryCreated = UR TRUE;
        (void)new UTL_FileSourceDir("DIR.TXT");
    }
}
// destructor
UTL_FileSource::~UTL_FileSource()
        // delete stuff
       delete[] theFileName;
        // unlink from list
    if( head == this )
        head = this->next;
    else
    {
```

```
UTL_FileSource * it = head;
           while( it )
                  if( it->next == this )
                         it->next = this->next;
                         it = 0;
                  }
                  else
                         it = it->next;
           }
   }
// Find the object corresponding to the given filename, which is currently
// accessible
// RETURNS: pointer to object, or null
UTL FileSource * UTL_FileSource::find(const char *filename)
       UTL FileSource * it = head;
while( it && ! it->isOne(filename) )
              it = it->next;
   if( it && ! it->isAccessible() )
       it = 0;
       return it;
}
//----
// Determine whether the given filename refers to this object.
// The base class function does a case-insensitive comparison of the given
// filename with the basic name of the object.
// Override this function for classes which respond to more than one filename.
UR_BOOLEAN UTL_FileSource::isOne(
               const char * filename // filename to compare
    const char *p1 = filename;
   while( *p1 && *p2 )
        if( ! filenameCharsMatch(*p1++,*p2++) )
           return UR FALSE;
    return ( *p1 || *p2 ) ? UR_FALSE : UR_TRUE;
                                             _____
// Extract the title of the file.
// Base class prints the filename.
void UTL_FileSource::printTitle(
   UTL FileUser & dest, // destination object
    int optionCount,
   const char *options[],
   const char *filename
{
    (void) optionCount;
    (void)options;
    (void) filename;
    dest.puts(theFileName);
// Compare two filename characters, to see if they match.
// Case is ignored, and backslash equals slash.
// RETURNS: UR_TRUE if the characters are equivalent, else UR_FALSE
UR_BOOLEAN UTL_FileSource::filenameCharsMatch(
                             char cl, // first character
                             char c2 // second character
    if( ( toupper(c1) == toupper(c2) )
|| ( c1=='\\'&&c2=='/')
```

```
|| ( c2=='\\'&&c1=='/') )
        return UR_TRUE;
    else
        return UR FALSE;
}
//-----
// Checks a file name to see if it matches the pattern baseName###.baseExt and extracts
// the number. Case is ignored, and backslash equals slash.
// RETURNS: UR_TRUE if there is a match.
UR BOOLEAN UTL FileSource::filenameMatch
    Const char * basename, // A base file name.
const char * filename, // A file name to check.
unsigned & number // Place to store the number.
{
                                       // Final return value.
// Pointer to base file name.
    UR_BOOLEAN result = UR_TRUE;
    const char * b = basename;
    const char * p = filename;
                                       // Pointer to file name under test.
    unsigned n = 0;
                                       // Number of digits collected.
                                       // Numeric digits collected from name.
    char digits[16];
    // Skip leading slashes, so filename is relative to root directory. while( *p == '\\' | | *p == '/' )
        p++;
    // Compare all the characters up to the dot or null
    while( *b && *b != '.' && result )
    {
        if( ! filenameCharsMatch(*p++,*b++) )
             result = UR_FALSE;
    }
    // If OK so far, gather digits from the given name \,
    while( *p && (*p != '.') && n < sizeof(digits)-1 && result )
        if( (*p < '0') || (*p > '9' ) )
            result = UR FALSE;
        else
             digits[n++] = *p++;
    // If still OK, see if the extension matches
    while( *b && result )
        if( ! filenameCharsMatch(*p++,*b++) )
             result = UR_FALSE;
    }
    // If any digits were collected then convert to a binary number
    // otherwise no match is found.
    if( n && result )
        digits[n] = 0;
        number = atoi( digits );
    return result;
// Checks whether the file is accessible. The file is not accessible if it
// requires a password which has not been entered. It is also not accessible
// if its menu file, or any menu file in the chain, is inaccessible (i.e., each
// file inherits the accessibility of its menu structure).
// RETURNS: UR TRUE if the file's access requirements are met
UR_BOOLEAN UTL_FileSource::isAccessible(void)
    UR_BOOLEAN returnValue = UR_TRUE;
    if( accessLevel == FACTORY_LEVEL )
    {
         if( ! MMI_Application::find()->isFactoryServiceEnabled() )
             returnValue = UR_FALSE;
    }
```

```
if( returnValue && menuFileName && *menuFileName ) // if has a non-blank menu name
       UTL FileSource * menuFile = UTL_FileSource::find(menuFileName);
       if ( menuFile )
           if( ! menuFile->isAccessible() )
                                         // menu file not accessible, so neither are we
               returnValue = UR FALSE;
       else
           returnValue = UR_FALSE; // menu doesn't exist, so neither does this file
   return returnValue;
// Delete all UTL FileSource objects
void UTL_FileSource::deleteAll(void)
       while( head )
       delete head;
}
Listing 5: UTL WebPage.h
                    ***************
 * Copyright (C) General Electric Co. GE Confidential and Proprietary
#ifndef _UTL_WEBPAGE_H_
#define _UTL_WEBPAGE_H_
#include "UTL_FileSource.h"
#define MAX HTML TABLE COLS 40 // maximum number of columns in an HTML table
// Web page class -- all web pages derive from this class.
// <BR> Key points:
// <UL>
         Subclasses can override "get", but they shouldn't. The "get" function
// <LI>
         sets up HTTP headers and calls the "printHTML" function, which most
//
         subclasses also shouldn't override.
//
         Subclasses should generally override the "getBody" function, providing
// <LI>
         HTML data for the part of the web page between the start and end
11
11
         of the page body.
         It's not a bad idea to learn some HTML if you're designing pages, but you can also use Front Page, Visual Interdev, etc. to design the
// <LI>
11
         page, then cut-and-paste the HTML into your code.
         If your page uses tables, use the UTL WebPage::Table class, it works well. In order to have your web page show up in a menu, specify the
// <LI>
// <LI>
         filename of the UTL_WebMenu object when constructing the page.
// <LI>
         You may specify an access level when constructing the web page, so
//
         that, for example, your page shows up only when factory service
11
         is enabled on the front panel.
// </UL>
class UTL_WebPage : public UTL_FileSource
public:
       UTL_WebPage(const char*filename,const char *aMenuFileName="", DB_SECURITY_LEVEL
anAccessLevel=NO LEVEL);
    ~UTL WebPage();
       virtual void get( UTL_FileUser & dest,int optionCount, const char *options[],const
char *filename);
    virtual void printHTML(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename);
    virtual void printPageHeading(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename);
protected:
    // HTML Table class, for use in the getBody function of a UTL_WebPage
```

```
// subclass. Create one on the stack, and it will automatically
    // wrap up with the appropriate HTML commands when it de-scopes.
   // You can also terminate a table with "end()" in order to start
   // a new one using the same object (perhaps calling "setWidth" to
// change the width).
   // <BR>
   // The startCell, startHeadingCell and startBannerCell functions start
   // different types of table cells. The class tracks column usage so
   // it will start new rows as required.
   // <BR>
   ^{\prime\prime} The remaining public functions set font styles. Font styles are valid ^{\prime\prime} for the remainder of the current cell, and are in general only changed
    // right after starting a cell, so the entire contents of the cell have the
    // same font.
class Table
   public:
       Table( int aNumCols, UTL_FileUser & dest );
       void setWidth( int aPercentWidth );
       void startTable(int aNumCols=0);
       void end(void);
       void startCell(const char * align="center", const char * bgColor=0, int colspan=1,
int rowspan=1);
       void startHeadingCell(const char * align="center");
       void startBannerCell(void);
       void setFontStyle(int size=3, const char *color="black");
       void setFontBold(UR_BOOLEAN onOff = UR_TRUE);
       void setFontItalic(UR BOOLEAN onOff = UR_TRUE);
   private:
       void nextRow(void);
       void endFont(void);
       UR_BOOLEAN inTable; // true if between start and end of table
       UR BOOLEAN inRow; // true if between start and end of row
       UR BOOLEAN inColumn;
                               // true if between start and end of column
                               // true if between start and end of font definition
// true if showing bold text
       UR_BOOLEAN inFont;
       UR BOOLEAN isBold;
       UR BOOLEAN isItalic;
                               // true if showinf italic text
                           // number of columns in the table
       int numCols;
       const char * bgcolor;
                              // background colour
       const char * fontcolor; // font colour
       UTL FileUser & dest;
                               //lint !e1725 destination for output
                               // width of table, in percent
       int percentWidth;
       unsigned short usedCols[MAX HTML TABLE COLS]; // to record pre-allocated columns
for multi-row cells
                               // number of column currently being shown (-1 if none yet)
       int columnNumber;
   void linefeed(UTL_FileUser & dest);
    virtual const char * getBackgroundColor(void);
       virtual void getHeader(UTL_FileUser & dest,int optionCount, const char
*options[]);
    // Get the html body text -- sub-classes must define this function.
       virtual void getBody(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename) = 0;
       virtual void printTitle(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename);
    // Convert Futaba character set to ISO for web browser.
    // RETURNS: dest, so you can use it in "printf"
    char * webString( char*dest, const char*src );
};
// Web page class for menus, which allow users to pick other web pages
// from a list. The "get" function prints a standard layout, including
// all the titles for the web pages which specified the particular
class UTL_WebMenu : public UTL_WebPage
public:
       UTL_WebMenu(const char*filename,const char *aMenuFileName="",const char *aTitle=0,
DB SECURITY LEVEL anAccessLevel=NO_LEVEL);
```

```
virtual void printTitle(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename);
virtual void getBody(UTL_FileUser & dest,int optionCount, const char
*options[],const char *filename);
private:
   const char *title; // menu title
#endif
Listing 6: UTL WebPage.cpp
* Copyright (C) General Electric Co. GE Confidential and Proprietary
#include "UTL_WebPage.h"
#include "UTL_FileUser.h"
#include "UTL StaticFile.h"
#include "DB_Text.h"
#include "DB_IPAddress.h"
#include "DB UINT16.h"
#include "SYS Product.h"
#include <stdio.h>
#include <assert.h>
// These data items are used in the standard page banner
extern DB Text Relay_Name;
extern DB_UINT16 Product_Version;
extern DB_IPAddress IP_Address;
// GIF file with the GE Power Management logo, used in the standard page banner
const char powerManagementLogo[] = {
 "\x99\xFF\xFF\x66\xFF\xFF\x33\xFF\xFF\x00\xFF\xCC\xFF\xFF\xCC\xCC\xFF\xCC\x99\xFF\xCC\x66"
"\xFF\xCC\x33\xFF\xCC\x00\xFF\x99\xFF\x99\xCC\xFF\x99\x99\x66\xFF\x99\x33\xFF"
"\x99\x00\xFF\x66\xFF\x66\xCC\xFF\x66\x99\xFF\x66\x66\xFF\x66\x33\xFF\x66\x00\xFF\x33"
"\xFF\xFF\x33\xCC\xFF\x33\x99\xFF\x33\x66\xFF\x33\x33\xFF\x33\x00\xFF\x00\xFF\x00\xCC"
"\xFF\x00\x99\xFF\x00\x66\xFF\x00\x33\xFF\x00\x00\xCC\xFF\xCC\xFF\xCC\xFF\x99\xCC"
"\x33\xCC\xCC\x00\xCC\x99\xFF\xCC\x99\xCC\x99\xCC\x99\xCC\x99\x66\xCC\x99\x33\xCC\x99\x00"
"\xCC\x66\xFF\xCC\x66\xCC\x66\x99\xCC\x66\x66\x33\xCC\x66\x30\xCC\x33\xFF\xCC"
"\x33\xCC\xCC\x33\x99\xCC\x33\x66\xCC\x33\x33\xCC\x33\x00\xCC\x00\xFF\xCC\x00\xCC\xCC\x00"
"\x99\xCC\x00\x66\xCC\x00\x33\xCC\x00\x99\xFF\xFF\x99\xFF\x99\xFF\x99\xFF\x99\xFF\x66"
"\x99\xFF\x33\x99\xFF\x00\x99\xCC\xFF\x99\xCC\x99\xCC\x99\xCC\x99\xCC\x66\x99\xCC\x33\x99"
"\xFF\x99\x66\xCC\x99\x66\x99\x96\x99\x66\x33\x99\x66\x33\xFF\x99\x33\xFF\x99\x33\xCC"
"\x99\x33\x96\x99\x33\x66\x99\x33\x33\x99\x33\x00\xF\x99\x00\xCC\x99\x00\x99\x99"
"\x00\x66\x99\x00\x33\x99\x00\x06\xFF\xFF\x66\xFF\xCC\x66\xFF\x99\x66\xFF\x66\x66\xFF"
"\x33\x66\xFF\x00\x66\xCC\xFF\x66\xCC\x66\xCC\x99\x66\xCC\x66\xCC\x33\x66\xCC\x00"
"\x66\x99\xFF\x66\x99\xCC\x66\x99\x96\x96\x99\x33\x66\x99\x00\x66\xFF\x66"
"\x66\xCC\x66\x99\x66\x66\x66\x66\x66\x33\x66\x66\x33\xFF\x66\x33\xCC\x66\x33"
"\x99\x66\x33\x66\x66\x33\x33\x66\x33\x00\x66\x00\xFF\x66\x00\xCC\x66\x00\x99\x66\x00\x66"
"\x66\x00\x33\x66\x00\x00\x33\xFF\xFF\x33\xFF\xCC\x33\xFF\x99\x33\xFF\x66\x33\xFF\x33\x33"
```

"\xFF\x00\x33\xCC\xFF\x33\xCC\xCC\x33\xCC\x99\x33\xCC\x66\x33\xCC\x33\xCC\x00\x33\x99" "\xFF\x33\x99\xCC\x33\x99\x99\x33\x99\x66\xCC" "\x33\x66\x99\x33\x66\x66\x33\x66\x33\x66\x00\x33\x35\xFF\x33\x33\xCC\x33\x33\x99\x33" "\x33\x66\x33\x33\x33\x33\x33\x33\x00\xFF\x33\x00\xCC\x33\x00\x99\x33\x00\x66\x33\x00" "\x33\x33\x00\x00\x00\xFF\xFF\x00\xFF\xCC\x00\xFF\x99\x00\xFF\x66\x00\xFF\x33\x00\xFF\x00" "\x00\xCC\xFF\x00\xCC\xCC\x00\xCC\x99\x00\xCC\x66\x00\xCC\x33\x00\xCC\x00\x00\x99\xFF\x00" "\x99\xCC\x00\x99\x99\x00\x99\x66\x00\x99\x33\x00\x99\x00\x06\xFF\x00\x66\xCC\x00\x66" "\x99\x00\x66\x66\x00\x66\x33\x00\x66\x00\x00\x33\xFF\x00\x33\xCC\x00\x33\x99\x00\x33\x66" "\x08\xFF\x00\xAF\x09\x1C\x48\xB0\xA0\xC1\x83\x08\x13\x2A\x5C\xC8\xB0\xA1\xC3\x83\x2B\xAC" "\x3C\xBC\xC6\x0A\xC0\x8A\x89\x18\x33\x6A\x54\xB8\x22\x22\x41\x56\x56\x00\xB0\xAA\x28\x71" "\x05\x00\x56\x02\x05\xFA\x38\xF2\xE2\xC6\x97\x30\x1B\xAE\x60\x35\x10\x25\xC5\x8E\x03" $"\x4D\xD2\xBC\x16\x48\x44\x40\x92\x31\x83\x0A\x7D\x18\xC8\x4A\xC8\x93\x11\x57\x0E\x5D\xCA"$ $"\xB4\xA9\xD3\xA7\x1B\x4F\x22\x25\x78\xF4\x24\x4D\x93\x52\x17\x56\x54\x2A\xD0\x8A\xCB\x88"$ "\xB9\xF0\xE8\x4E\xB9\x5C\xD5\x66\x4C\x5B\x90\x95\xCA\x94\x81\x46\xEA\x15\xAC\x90\x2E\xCF" "\xC0\x21\x07\x12\xD6\xBA\xD8\x60\x63\xBD\x90\x23\x4B\x9E\x4C\xB9\xB2\xE5\xA5\x45\x3B\x8E". "\x55\x79\xB9\x73\x57\x9F\x02\x57\x26\x5E\xC1\x15\x40\xE0\xB3\x04\x39\x7B\x86\x7A\x36\x30" "\x55\xD3\x15\x79\xFA\x44\x19\x92\xEE\xD8\xD5\x83\x6D\x4E\xF4\x8A\xBB\xB7\xEF\xDE\x72\xD9" "\xB2\x95\x28\x50\xB8\x71\x97\xA1\x70\x86\xFD\xFD\x90\x2D\x72\x84\x55\x8D\xEB\x66\x6C\x38" "\xE1\xE3\xCA\xC2\x77\x56\x94\x2A\x71\xFB\x49\x23\xC3\x5F\xFF\x8F\xDD\xB1\x27\xDA\x8B" "\x55\x2F\xE2\x5C\x8E\x15\xB6\x4F\xDE\x26\xCD\x82\xBC\xB8\xDC\x74\x4F\xCD\x12\x4D\x5F\xE3" "\xCD\x9B\xB9\x7F\x83\xB3\x3C\xB2\x86\x80\x04\x0E\x68\x60\x81\x08\x1E\xA8\x60\x82\x0C\x2E" "\xE8\x60\x83\xFF\x45\x28\xE1\x84\xAB\xF5\xC4\x55\x70\x5E\xB5\x97\x17\x85\x41\x99\x67\x50" "\x51\x05\x15\x95\x17\x69\x1C\xBE\xD4\x13\x71\x37\xC5\x66\x1A\x89\x72\xA1\x48\xD5\x86\x25" -"\x36\x84\xD2\x74\x1F\xB5\xD5\x53\x75\x02\x99\x14\xC8\x73\x31\x36\xA4\x5A\x72\x3B\xD9\x74" "\xE3\x56\xB2\xE5\xA4\x5F\x8F\x18\x19\xA5\x98\x6B\x1F\xAA\x44\x17\x91\x48\xC2\x54\x55\x4A" "\x55\xD1\x65\x61\x94\x58\x66\xA9\xE5\x6A\x7E\xA5\x84\x63\x92\x7E\xD9\x63\x76\x37\xB1" "\x45\x53\x70\x28\x89\x48\xA5\x61\x1D\xF9\xD5\x56\x9B\x6D\xCE\xE6\x53\x44\x5E\xA9\x24\x52" "\x45\xA4\x89\xE9\x54\x74\x7A\x52\x04\x57\x5D\x3C\x7A\x54\xDB\x35\x16\xAD\x14\x5F\x91\x21" "\x9D\xD6\xE7\x9E\x66\xFA\x69\x5C\x71\x52\x69\x26\xE9\x97\x5E\x2A\x96\xD2\x98\xDC\xBD\x96" "\x15\x5A\x91\x3A\x84\x92\x47\x16\xB6\x66\x51\x62\x24\x25\xCA\xE9\x8E\xE8\x91\xA6\xD3\x56"

```
"\x57\x96\xF5\xDD\x9B\x33\xF1\xFF\x67\xDA\x49\x76\x32\x29\x9B\x74\x8D\x46\x27\x1D\x8E\x38"
"\xDD\xE7\x27\x48\x22\xB5\x4A\x13\x7C\x67\x99\x24\x9B\x9B\x2B\xB9\x77\x91\x7E\xBC\x99\x87"
"\x93\x4E\x1E\xEA\xD7\xD6\x96\x14\x01\x46\xED\xB5\x07\x05\xA8\xED\x23\xDB\x76\xCB\xED\xB7"
"\xDE\x86\x0B\xEE\xB8\xE2\x96\x3B\x2E\xB6\xE8\xA6\xAB\xEE\xBA\xEC\xB6\xEB\xEE\xBB\x31\x01"
"\x0B\xDE\x48\x99\x99\x05\x23\xBC\x05\xF1\xF5\x13\x8D\x65\xBA\x88\xEF\x79\x1A\xE9\xEB\x2E"
"\x50\xF9\xEA\xA8\xD8\x8C\x05\xA1\xF6\xEE\x49\x8A\x2D\xAB\x28\x90\x47\x12\xA4\x30\xBB\x75"
"\x4A\x8C\x95\x9D\xC3\x46\x1C\xE2\xBD\xD8\x22\x9C\x2F\x77\x28\xE1\xA9\xAF\x55\xFE\xA6\x3B"
"\xEB\xC4\x8E\xCE\xB6\x12\x48\x97\xD6\xD4\x1F\xC5\xD5\x19\xBB\x9F\xA8\x45\x02\x9C\xA3\x92"
"\xED\xC6\x96\x13\x71\x57\x12\xA9\x14\x4E\x90\xC2\xCB\x0A\x72\xC1\x05\xFA\xE7\x67\x8B\x62"
"\x4B\x62\x46\x2F\xFF\x7B\x6A\x75\x89\x15\xD4\xB4\xD3\x0D\x47\x9A\x54\xB5\xFC\x52\xAD\xF5"
"\xD6\x5C\x77\xED\xF5\xD7\x0D\x01\x6B\x5C\x5E\xBA\x3E\x7A\xB0\x59\xA0\x65\x89\x26\xAE\x35"
"\xE1\xFA\xDD\x6B\x40\xCE\x37\xD3\xD0\x67\xCE\xC4\x53\xAA\x69\x8D\x95\xF4\x52\xD1\x5D\xFF"
"\xD8\x28\x9A\x8B\x0E\x57\xB3\x57\x61\x79\x75\xA7\x48\x1E\xC5\x16\x35\x65\xDE\x69\x8C\x2A"
"\xD0\xCE\x2D\xE4\xE1\xCE\xC5\x9D\xB6\x23\x5A\xB5\x85\x15\x1E\xA1\x1C\xAB\xD5\x1E\x8F\x1F"
"\x3A\x27\xA9\xC0\x40\x0F\x54\x95\x52\x0C\x73\x4A\x93\x77\xF4\xA1\x1C\x59\xE3\xA8\x1B\x77"
"\x95\xDB\x1A\xAB\x1B\x5D\x72\xC2\xB5\x9E\x29\x43\xDB\xD9\x36\x6D\x84\x79\x45\x3E\x10\xE0"
"\x33\x6F\xCA\x90\xE1\x3F\x1B\x9C\x21\x71\xA4\x11\xDE\x55\x9B\xCF\x0F\xBD\x72\x47\xCC\x17"
"\x05\x5E\xC5\xFB\x95\x77\xF7\xD0\x25\x8D\xDA\xF6\xDB\x46\x45\xE8\xEE\x5A\x21\x6E\xA8"
"\x51\xF4\x21\x26\x2A\x4D\x36\x79\xE9\x91\x90\x67\xF5\x17\xB2\xCA\x98\x13\x09\x5F\x77\x34"
"\x13\xD4\x1E\x52\x63\x17\xEF\x36\x8C\xFC\x31\x98\x7B\xCE\x87\x96\x95\xC5\xAF\x75\x6E\xBA"
"\x19\xE7\x36\xD7\xAB\xB3\x14\x6E\x6E\x81\x59\x0E\x7C\x60\xB4\xB6\xDC\x41\x8A\x76\x6B\x89"
"\x0B\xB3\x44\x65\x91\x60\xA1\x46\x50\x48\xD1\x9D\x4E\x38\x65\x95\xFA\x88\xE6\x22\x18\xC2"
"\xDC\xAF\xB6\xD3\xB9\x12\xC5\x27\x6B\x60\x8B\xA1\x0C\x67\xE8\xB5\x47\x04\xE1\x86\x38\xCC"
"\xA1\x0E\x77\xC8\xC3\x1E\xFA\xF0\x87\x3E\xA4\xA1\x10\x0C\x87\x48\xC4\x22\x1A\xF1\x88\x48".
  \x1C\x48\x40\x00\x00\x3B"
static UTL_StaticFile * logoFile = 0;
const char URWellConnected[] = {
  "\x47\x49\x46\x38\x39\x61\x69\x00\x5B\x00\xF7\x00\x00\x00\x00\x33\x00\x00\x66\x00"
"\x00\x99\x00\x00\xCC\x00\x00\xFF\x00\x00\x00\x33\x00\x33\x33\x00\x66\x33\x00\x99\x33\x00"
"\xCC\x33\x00\xFF\x33\x00\x06\x00\x66\x00\x66\x06\x06\x00\x99\x66\x00\xCC\x66\x00\xFF"
"\x66\x00\x00\x99\x00\x33\x99\x00\x66\x99\x00\x99\x99\x00\xCC\x99\x00\xFF\x99\x00\xCC"
"\x00\x33\xCC\x00\x66\xCC\x00\x99\xCC\x00\xCC\xCC\x00\xFF\xCC\x00\xFF\x00\x33\xFF\x00"
"\x66\xFF\x00\x99\xFF\x00\xCC\xFF\x00\xFF\xFF\x00\x00\x00\x33\x33\x00\x33\x66\x00\x33\x99"
"\x00\x33\xCC\x00\x33\xFF\x00\x33\x00\x33\x33\x33\x33\x66\x33\x33\x99\x33\x33\xCC\x33"
"\x33\xFF\x33\x33\x00\x66\x33\x33\x66\x33\x66\x33\x99\x66\x33\xCC\x66\x33\xFF\x66\x33"
"\x00\x99\x33\x99\x33\x66\x99\x33\x99\x33\xCC\x99\x33\xFF\x99\x33\x00\xCC\x33\x33"
"\xCC\x33\x66\xCC\x33\x99\xCC\x33\xCC\xCC\x33\xFF\xC33\x00\xFF\x33\x56\xFF"
"\x33\x99\xFF\x33\xCC\xFF\x33\xFF\xF33\x00\x66\x33\x00\x66\x66\x00\x66\x99\x00\x66"
"\xCC\x00\x66\xFF\x00\x66\x00\x33\x66\x33\x66\x33\x66\x99\x33\x66\xCC\x33\x66\xFF"
```

"\x33\x66\x00\x66\x66\x33\x66\x66\x66\x66\x66\x99\x66\x66\xCC\x66\x66\xFF\x66\x66\x99\. "\x66\x33\x99\x66\x66\x99\x66\x99\x66\xCC\x99\x66\xFF\x99\x66\x00\xCC\x66\x33\xCC\x66" "\x66\xCC\x66\x99\xCC\x66\xCC\x66\xFF\xCC\x66\x00\xFF\x66\x33\xFF\x66\x66\x99" "\xFF\x66\xCC\xFF\x66\xFF\x66\x00\x00\x99\x33\x00\x99\x66\x00\x99\x00\x99\xCC\x00" "\x99\xFF\x00\x99\x00\x33\x99\x33\x99\x66\x33\x99\x95\x33\x99\xCC\x33\x99\xFF\x33\x99" "\x00\x66\x99\x33\x66\x99\x66\x66\x99\x99\x66\x99\xCC\x66\x99\xFF\x66\x99\x00\x99\x33" "\x99\x99\xCC\x99\xCC\x99\xFF\xCC\x99\x00\xFF\x99\x33\xFF\x99\x66\xFF\x99\x99\xFF\x99" "\xCC\xFF\x99\xFF\xFF\x99\x00\xCC\x66\x00\xCC\x99\x00\xCC\xCC\x00\xCC\xFF" "\x00\xCC\x00\x33\xCC\x33\x33\xCC\x66\x33\xCC\x99\x33\xCC\xCC\x33\xCC\xFF\x33\xCC\x00\x66" "\xCC\x33\x66\xCC\x66\xCC\x99\x66\xCC\x66\xCC\xFF\x66\xCC\x33\x99\xCC\x33\x99\xCC" "\x66\x99\xCC\x99\x99\xCC\xCC\x99\xCC\xF\x99\xCC\x00\xCC\xCC\x33\xCC\xCC\x66\xCC\xCC\x99" "\xCC\xFF\xFF\xCC\x00\x00\xFF\x33\x00\xFF\x66\x00\xFF\x99\x00\xFF\xCC\x00\xFF\x00\xFF" "\x00\x33\xFF\x33\x33\xFF\x66\x33\xFF\x99\x33\xFF\xCC\x33\xFF\x66\xFF\x00\x66\xFF\x33" "\x66\xFF\x66\xFF\x99\x66\xFF\xCC\x66\xFF\xFF\x66\xFF\x00\x99\xFF\x33\x99\xFF\x66\x99" "\xFF\x99\x99\xFF\xCC\x99\xFF\x99\xFF\x00\xCC\xFF\x33\xCC\xFF\x66\xCC\xFF\x99\xCC\xFF" "\x08\xFE\x00\x01\x08\x1C\x48\xB0\xA0\xC1\x83\x08\x13\x2A\x5C\xC8\xB0\xA1\xC3\x87\x10\x23" "\x4A\x9C\x48\xB1\xA2\xC5\x8B\x18\x33\x6A\xDC\xC8\xB1\x23\xC3\x00\x05\x42\x16\x08\x10\x71" "\x40\x01\x01\x09\x05\x88\x2C\x70\x10\xA4\x48\x94\x1E\x27\xAA\x0C\x49\x40\x22\x01\x96\x29" "\x45\x0E\x40\x78\x53\x24\xC9\x98\x25\x75\x4A\x2C\x50\x33\x61\xCF\x93\x05\x8F\xC2\x04\x6A" "\xD1\x64\xC8\x9F\x0A\x5D\x2E\x6D\xB9\x12\xAA\x41\x97\x4F\x99\x6A\xDD\xCA\xB5\xAB\xD7\xAF" "\x60\xC3\x8A\x1D\x4B\xB6\xAC\xD9\xB3\x68\xD3\xAA\x0D\x3B\xE0\xE8\x4A\x91\x04\x76\x22\x74" "\xFA\x96\x00\x01\x92\x02\xDC\xBE\xA5\x49\x40\x80\xD5\xB5\x58\x71\x42\x34\xF9\x97\xE0\x4C" "\xA2\x46\x57\xA6\x3D\x2C\x17\xE2\x4D\x85\x74\xFB\xDA\x5D\x39\x75\x2D\x00\xB8\x41\x2B\x27" "\x7D\x79\x90\xB1\xE5\x96\x6D\xF7\xC6\x2D\x5C\x50\x80\xE9\xD3\xA8\x4D\x93\x06\x90\xFA\xF4\" $"\xE7\xD7\xB0\x63\xCB\x9E\x4D\xBB\xB6\xED\xDB\xB8\x73\xEB\xDE\xCD\xBB\xB7\xEF\xDD\x6F\x57"$ "\x23\x9C\xB9\xFA\x30\xD2\x81\x79\xAB\x7E\xA6\xDB\xD8\x21\xC8\xA2\x3C\x39\x1F\x74\xBB\x56" "\x69\x44\x90\xCD\x0D\x2A\xEF\x4C\x59\xAD\x75\x88\x2A\x35\xFE\x13\xDC\x0E\x20\x80\x00\xBA" "\x44\x85\x8F\x25\xEF\x30\x7C\xC2\xC0\x86\x31\xC3\x3E\x6A\x73\x64\x4E\x9A\x57\xD9\xAB\xA5" "\xAB\xBE\xA0\xC9\xEC\xA5\x09\x15\x5D\x48\xE2\xA5\x75\xD4\x00\x9A\x25\x77\xDC\x7B\x6F\x15"

```
"\x28\x90\x71\x0E\xFE\x26\xE1\x84\x14\x56\x68\xE1\x85\x18\x66\xA8\xE1\x86\x1C\x76\xE8\xE1"
"\x87\x20\x86\x28\xE2\x88\x24\x96\x68\xE2\x89\x28\xA6\xA8\xE2\x8A\x64\x5C\x7A\xCE"
"\x1D\x55\x58\x60\x7C\xE9\x75\x17\x6C\x2B\x01\xB8\x90\x4A\xD0\xE5\xE7\x93\x41\xD4\x59\x76"
"\x58\x8F\x0E\x3D\x96\x10\x73\x08\xAD\x44\xE4\x59\xDF\x3D\xE4\x1E\x42\xF0\x0D\x18\xD2\x5A"
"\x43\x0E\x25\xD8\x41\x74\x39\x98\xE3\x5A\xFA\x31\xF4\x24\x55\xF8\x25\x29\x5F\x5A\x48\x46"
"\x84\x58\x62\x04\x5E\xE5\x56\x7F\x60\x45\x39\x98\x7D\xC3\xED\x25\xE7\x00\x6C\x8A\xD5\xA4"
"\x73\x05\xE8\x38\x90\x8C\x03\x65\x19\x5B\x95\x11\x19\x19\x67\x48\xD9\x75\x69\xA0\x74\x4E"
"\x2E\x78\x90\x7E\x80\x7E\xA6\x18\x44\xCF\x41\x26\xE0\x66\x69\x7E\x43\x2E\x45\xF5"
"\xA8\x8F\x53\x7E\x86\x55\x84\xF1\xC1\x39\x17\xA2\x06\x35\x6A\x19\x9F\xF7\xE9\x89\x9C\xA1"
"\xD6\xD9\xE2\xAF\xC0\x06\x2B\xEC\xB0\xC4\x16\x6B\xEC\xB1\xC8\x26\xAB\xEC\xB2\xB8\x05\x04"
 "\x00\x3B"
static UTL StaticFile * urGifFile = 0;
// The default web page (main menu)
static UTL_WebMenu * mainMenuFile = 0;
// Constructor -- creates a web page
UTL WebPage::UTL_WebPage(
                    const char*filename,
                                     // filename by which to find this file
                    const char *aMenuFileName, // filename of menu in which to
include this file
                    DB SECURITY LEVEL anAccessLevel
                                                // access level
            UTL FileSource(filename, anAccessLevel)
{
   menuFileName = aMenuFileName;
   if( !logoFile ) // Use logoFile pointer to see if the whole group has been constructed
      logoFile = new UTL StaticFile("/logo.gif", (unsigned char*)&powerManagementLogo,
sizeof(powerManagementLogo);
      assert( logoFile );
      urGifFile = new UTL_StaticFile("/URWellConnected.gif", (unsigned
char*)&URWellConnected, sizeof(URWellConnected));
      assert( urGifFile );
      mainMenuFile = new UTL_WebMenu("/default.htm","","Main Menu");
      assert ( mainMenuFile );
   }
}
// Destructor
UTL WebPage::~UTL WebPage()
// Get the contents of the web page, including HTML header info.
void UTL WebPage::get(
      UTL_FileUser & dest,
                         // where to send the data
                     // number of optiosn
   int optionCount,
                            // options -- ignored at this livel, but may be used in
   const char *options[],
other functions
   const char *filename // filename - ignored, since web pages use options instead
{
   (void) filename;
      // Format the header part
      dest.puts( (char*)
      "HTTP/1.0 200 OK \r\n"
```

```
"Server: GE Industrial Systems UR\r\n"
       "Content-Type: text/html\r\n"
       "\r\n");
   printHTML( dest, optionCount, options, filename );
}
// Print the non-header part of the page. Override if you don't want the normal
// head, body layout (for a frameset, for example).
void UTL WebPage::printHTML(
       UTL FileUser & dest,
                                 // where to send the file
                             // number of options
       int optionCount,
   const char *options[],
const char *filename
                               // options
                             // filename, in case it matters
{
   dest.puts(
              "<HTML>\n"
              "<HEAD>\n"
              "<TITLE>" );
       char s[100], a[100];
       UR_UINT16 c;
    (void) Relay Name.get((char*)s);
    (void) Relay Name.toAscii(&c, (char*)a, (void*)s);
    dest.puts(a);
    dest.puts(" ");
       printTitle(dest, optionCount, options, filename);
       dest.puts( "</TITLE>\n");
       getHeader(dest, optionCount, options);
       dest.puts( "</HEAD>\n" );
       // format the body
dest.printf( "<BODY BGCOLOR=%.50s>\n", getBackgroundColor());
   printPageHeading( dest, optionCount, options, filename );
       getBody(dest, optionCount, options, filename);
       dest.puts( "</BODY></HTML>\n" );
}
// Print a banner or heading at the top of the web page. Override if you don't
// want the normal heading (unless you're already overriding printHTML, from which
// this function is called).
void UTL WebPage::printPageHeading(
       UTL_FileUser & dest,
                                 // where to send the file
                             // number of options
       int optionCount,
                               // options
    const char *options[],
   const char *filename
                             // filename, in case it matters
{
    (void)optionCount;
    (void)options;
    (void) filename;
                          // Number of characters printed to buffer.
    UR UINT16 count;
                         // Buffer for use with getFormattedLine.
    char buffer[256];
    UR UINT16 versionNumber;
    UR BOOLEAN isHomePage = isOne("default.htm");
    Table t(4,dest);
    t.startCell("left", "black", 1, 2);
    dest.puts(
       "<IMG SRC=\"/logo.qif\" ALT=\"GE Power Management Logo\" TITLE=\"GE Power
Management Logo\">\n"
       );
    t.startCell("left", "silver");
    t.setFontBold();
    dest.printf("   %s<BR>", SYS_Product::find()->name );
    (void)Product_Version.get(&versionNumber);
(void)Product_Version.toAscii(&count, (char*)&buffer, &versionNumber );
    dest.printf("   Revision %s", buffer);
    t.startCell("right", "silver");
    t.setFontItalic();
```

```
dest.puts("Relay Name: ");
t.setFontItalic(UR_FALSE);
   t.setFontBold();
   Relay_Name.getFormattedValue(&count; buffer);
   dest.puts(buffer);
   t.setFontBold(UR_FALSE);
   dest.puts("    ");
dest.puts("<BR>");
   t.setFontItalic();
dest.puts("IP Address: ");
   t.setFontItalic(UR_FALSE);
   t.setFontBold();
   IP_Address.getFormattedValue(&count, buffer);
   dest.puts(buffer);
   dest.puts("     ");
   t.startCell("right", "black", 1, 2);
   dest.puts(
      "<IMG SRC=\"/URWellConnected.gif\" ALT=\"UR Logo\" TITLE=\"UR Logo\">\n"
   t.startCell("center", "white", 2);
   t.setFontStyle(5);
   t.setFontBold();
   printTitle(dest, optionCount, options, filename);
   dest.puts("<BR>");
   if( ! isHomePage )
      t.setFontBold(UR FALSE);
      t.setFontStyle(3);
      dest.puts("<A HREF=default.htm>Click Here For The Main Menu</A>\n");
   t.end();
   dest.puts("<BR>\r\n");
}
// Get a background colour for the page. The default is "silver" - subclasses
// may override.
const char * UTL_WebPage::getBackgroundColor(void)
{
   return (const char *) "silver"; // the default
// Get the HTML header text. The base class does nothing here.
void UTL_WebPage::getHeader(
                           // output to here
// number of options
     UTL_FileUser & dest,
     int optionCount,
                       // options
   const char *options[]
)
   (void)optionCount;
   (void) options;
   (void) dest;
// Write a title -- default is just the file name, but subclasses should override.
void UTL WebPage::printTitle(
                           // where to send the file
     UTL_FileUser & dest,
                        // number of options
     int optionCount,
   const char *options[],
const char *filename
                          // options
                        // filename, in case it matters
   (void)optionCount;
   (void) options;
   (void) filename;
   dest.puts( theFileName );
```

```
// Convert Futaba character set to ISO for web browser.
// RETURNS: dest, so you can use it in "printf"
char * UTL WebPage::webString(
                            char*dest;
                                             // destination buffer -- make sure it's
big enough
                                             // source string
                            const char*src
{
   char *p = dest;
   while( *src )
    ſ
       switch( 255 & (*src) )
       case 0x7f: // all pixels on
                                     // not perfect, but I guess it will do
           p += sprintf(p, "Ξ");
           break;
                  // degree
       case 0xDF:
           p += sprintf(p, "°");
           break;
       case 0x88: // micro
           p += sprintf(p,"μ");
           break;
       case 0x8e: // ohms
           p += sprintf(p, "Ω");
           break;
       case 0x8d: // phase symbol
   p += sprintf(p,"Φ");
           break;
       default:
           *p++ = *src;
           break;
       src++;
    *p = 0;
   return dest;
// Table constructor -- creates an HTML table, which will terminate on de-scoping.
// You should generally create this guy on the stack.
// EXAMPLE:
        SomeSubclass::getBody( ...
11
11
11
            UTL WebPage::Table t(2,dest);
            t.startBannerCell();
            printTitle(dest,optionCount,options,filename);
//
            t.nextRow();
            t.startHeadingCell();
            dest.puts("first column heading");
            t.startHeadingCell("left");
            dest.puts("second column heading");
//
            while ( some condition ) .
                (get next row data)
                t.nextRow();
                t.startCell();
                dest.printf("%d", some_value);
                t.startCell("left");
                dest.printf("%s", some_text);
            }
11
        }
                           ______
UTL WebPage::Table::Table(
                                             // number of table columns
                        int aNumCols,
                        UTL_FileUser & aDest // destination for the HTML output
    : dest(aDest)
{
    inTable = UR_FALSE;
    inRow = UR_FALSE;
```

```
inColumn = UR_FALSE;
   inFont = UR FALSE;
   isBold = UR FALSE;
   isItalic = UR FALSE;
   numCols = aNumCols > MAX_HTML_TABLE_COLS ? MAX_HTML_TABLE_COLS : aNumCols;
   bgcolor = 0;
   fontcolor = 0;
   percentWidth = 95;
   columnNumber = -1;
for( int i=0; i<MAX_HTML_TABLE_COLS; i++ )</pre>
      usedCols[i] = 0;
}
// Table destructor -- terminates HTML table, if one has started.
UTL WebPage::Table::~Table()
   end();
   bgcolor = 0;
   fontcolor = 0;
}
// Set the width of the table, in percent. Call this function before any of the
// other functions, to set the width different from the default (95%).
void UTL WebPage::Table::setWidth(
    int aPercentWidth // width of subsequently-started table (5-100 percent)
   percentWidth = aPercentWidth;
   if ( percentWidth > 100 )
     percentWidth = 100;
   if( percentWidth < 5 )</pre>
      percentWidth = 5;
}
// Start a table, terminating the previous one if it's started.
void UTL_WebPage::Table::startTable(
                  int aNumCols
                                  // number of table columns
{
   end();
   if( aNumCols > 0 )
     numCols = aNumCols > MAX_HTML_TABLE_COLS ? MAX_HTML_TABLE_COLS : aNumCols;
   dest.puts("<BR>\r\n");
   dest.printf("<TABLE width=%d%% align=center bgColor=#F0F0F0 border=2 borderColor=black
border=2 cellspacing=0 cellpadding=3>",
     percentWidth);
   inTable = UR_TRUE;
   columnNumber = -1;
for( int i=0; i<MAX_HTML_TABLE_COLS; i++ )</pre>
      usedCols[i] = 0;
}
// Terminate the table -- generally only call this function if you want to output
// some HTML before starting another table (otherwise you can rely on the destructor).
void UTL_WebPage::Table::end(void)
   endFont();
   if(inColumn)
      dest.puts("</TD>");
      inColumn = UR_FALSE;
   if(inRow)
      dest.puts("</TR>");
      inRow = UR_FALSE;
```

```
if(inTable)
       dest.puts("</TABLE>");
       inTable = UR_FALSE;
   }
}
// Start a row of cells, wrapping up the previous row, if any, and starting
// a table, if not already started.
void UTL WebPage::Table::nextRow(void)
   endFont();
   if(!inTable)
       startTable();
   if( inColumn )
       dest.puts("</TD>");
       inColumn = UR_FALSE;
   if( inRow )
      dest.puts("</TR>");
   dest.puts("\r\n<TR valign=center>");
   inRow = UR TRUE;
   columnNumber = -1;
// Start a column, wrapping up the previous one if any, and starting the table
// and/or row if necessary.
void UTL WebPage::Table::startCell(
                           // alignment ("center","left","right")
// background colour ("white","silver","yellow", etc.)
      const char * align,
const char * bgColor,
                           // number of columns to span (generally 1)
       int colspan,
                           // number of rows to span (generally 1)
       int rowspan
   int i;
   endFont();
   if(!inTable)
       startTable();
   if(!inRow)
      nextRow();
   if( inColumn )
       dest.puts("</TD>");
   // Find the columns which can hold our cell.
   // If this is a multi-row cell, reserve the columns it needs.
   // Expect screw-ups if the configuration is truly whacky, like a colspan cell
   // spanning over a previous-row rowspan cell.
   int colsToReserve = colspan;
                                  // if last row ended at end of row...
   if( (columnNumber+1) >= numCols )
                                   // ...start a new row
      nextRow();
   while( colsToReserve )
       if( ++columnNumber < numCols )</pre>
       {
          if( usedCols[columnNumber] )
              usedCols[columnNumber]--; // can't use this one, but absorb a
reservation
          else
              if( rowspan > 1 )
                 usedCols[columnNumber] = rowspan-1; // reserve the column for as
many rows as necessary
              colsToReserve--;
       1
       else
                           // end of the row
          nextRow();
```

```
colsToReserve = 0; // get out of here
   }
   if(rowspan > 1)
      short minReserved = 32767;
      for( i=0; i<numCols; i++ )
         if( usedCols[i] < minReserved )
            minReserved = usedCols[i];
      if( minReserved )
         for( i=0; i<numCols; i++ )</pre>
            usedCols[i] -= minReserved; // eliminate completely reserved rows
   inColumn = UR TRUE;
   dest.puts("<TD");
   if( colspan > 1 )
      dest.printf(" colspan=%d", colspan);
   if( rowspan > 1 )
      dest.printf(" rowspan=%d", rowspan);
   if( bgColor )
   dest.printf(" bgcolor=%s", bgColor);
dest.printf(" align=%s>", align);
void UTL WebPage::Table::startHeadingCell(const char * align)
{
   startCell(align,"silver");
   dest.puts("<FONT color=black size=4><STRONG>\r\n");
   isBold = UR_TRUE;
   inFont = UR TRUE;
}
// Start a "banner" cell, spanning an entire row, with special highlighting.
void UTL WebPage::Table::startBannerCell(void)
   endFont();
   if(!inTable)
      startTable();
   if( inColumn )
      dest.puts("</TD>");
   if( inRow )
      dest.puts("</TR>");
   dest.printf("\r\n<TR><TD align=center colspan=%d bgcolor=#483D8B><FONT color=white
size=5><STRONG>",numCols);
   columnNumber = numCols; // force next cell to new row
   inRow = UR_TRUE;
   inColumn = UR_TRUE;
   inFont = UR_TRUE;
   isBold = UR TRUE;
ì
// Turn off any special font formatting.
void UTL_WebPage::Table::endFont(void)
   if( isItalic )
      isItalic = UR FALSE;
      dest.puts("</EM>");
   if( isBold )
```

```
isBold = UR FALSE;
     dest.puts("</STRONG>");
  if( inFont )
     inFont = UR FALSE;
     dest.puts("</font>");
  }
}
// Change font style for remainder of this table cell
void UTL WebPage::Table::setFontStyle(
                       // size of font (normal is 3)
          int size,
          const char *color
                       // font colour
{
  if( inFont )
     dest.puts("</FONT>");
  inFont = UR TRUE;
  dest.printf("<FONT size=%d color=%s>", size, color );
// Turn bold text on or off for remainder of this table cell
void UTL_WebPage::Table::setFontBold(
     UR_BOOLEAN onOff // true for bold, false for noraml
  if( onOff )
     if(!isBold)
       dest.puts("<STRONG>");
     isBold = UR_TRUE;
  else
     if( isBold )
       dest.puts("</STRONG>");
     isBold = UR_FALSE;
}
// Turn italic text on or off for remainder of this table cell
void UTL WebPage::Table::setFontItalic(
     UR BOOLEAN onOff
                  // true for italic, false for noraml
  if( onOff )
     if( !isItalic )
    dest.puts("<EM>");
     isItalic = UR_TRUE;
  }
  else
     if( isItalic )
        dest.puts("</EM>");
     isItalic = UR_FALSE;
  }
}
// Constructor -- creates a web page for a menu
UTL WebMenu::UTL_WebMenu(
                const char*filename,
                              // filename by which to find this file
                const char *aMenuFileName,// filename of menu in which to include
this file
```

```
UTL_WebPage(filename,aMenuFileName,anAccessLevel)
{
   assert( aTitle );
   title = aTitle;
}
// Print the menu title
void UTL WebMenu::printTitle(
      UTL_FileUser & dest,
                             // where to send the file
                         // number of options
      int optionCount,
   const char *options[],
const char *filename
                           // options
                          // filename, in case it matters
{
   (void)optionCount;
   (void) options;
   (void) filename;
   dest.puts(title);
}
// Print the menu
void UTL WebMenu::getBody(
                             // where to send the data
      UTL FileUser & dest,
                         // number of options
      int optionCount,
   const char *options[],
const char *filename
                           // options
                          // filename, in case it matters
{
   (void) optionCount;
   (void) options;
   (void) filename;
   Table t(1,dest);
   t.startBannerCell();
   dest.puts("Select from the following options");
   t.startCell();
   t.setFontBold();
   t.setFontStyle(5);
   // Find all the pages which want to be in this menu, and put them in a table
UTL_FileSource * it = UTL_FileSource::getFirst();
   while( it )
   {
      const char *dummyOptions[1];
      if( isOne(it->getMenuFileName()) ) // Am I this guy's menu file?
          if( it->isAccessible() )
             dest.printf("<A HREF=%.200s>", it->getFileName() );
             it->printTitle(dest,0,dummyOptions,it->getFileName());
             dest.puts("</A><BR>\r\n");
      it = it->getNext();
   }
}
Listing 7: UTL FileUser.h
        *******************
 * Copyright (C) General Electric Co. GE Confidential and Proprietary
 * DESCRIPTION Generic file user class
#ifndef _UTL_FILEUSER_H_
#define _UTL_FILEUSER_H_
#include "SYS_Types.h"
```

```
// Generic file user class, to obtain data from UTL_FileSource objects.
// Subclasses override the sendFrame function to modify the mechanics
// involved in getting blocks of data where they have to go.
// <BR> Key functions are:
// <UL>
// <LI>
         printf - formatted, buffered print
         puts - buffered write of a string
write - block write
// <LI>
// <LI>
// <LI>
         flush - send any unsent information from the buffers
// </UL>
class UTL FileUser
public:
   int printf (const char * fmt, ...); //lint !e1916
       void puts( const char * txt );
       void write(unsigned char *buffer, UR_UINT16 length);
    void flush(void);
       UR_UINT16 getLength(void)
                                  // get the maximum buffer length
              return theLength;
                                      // get a buffer into which to format the
       unsigned char * getBuffer(void)
frames
              return theBuffer;
   virtual ~UTL FileUser();
protected:
       virtual void sendFrame(unsigned char *buffer, UR_UINT16 length) = 0;
       UTL FileUser(unsigned char *buffer, UR_UINT16 length);
                                  // points to a handy buffer for formatting messages
       unsigned char * theBuffer;
                                   // size of the handy buffer
       UR UINT16 theLength;
                              // number of characters waiting to be sent
    UR_UINT16 bufferedChars;
};
#endif
Listing 8: UTL_FileUser.cpp
                            ***********
 * Copyright (C) General Electric Co. GE Confidential and Proprietary
 * DESCRIPTION File user class
#include "UTL FileUser.h"
#include <assert.h>
// va_list is defined differently in visual C++ and GNU, so we need to tweak the
// code to match the compiler being used.
#ifdef WIN32
    // Definitions from the Visual C++ stdarg.h
   #undef va_start
#undef va_end
    \#define va_start(ap, v) (ap = (char*)&v + ((sizeof(v) + sizeof(int) - 1) &
\sim (sizeof(int) - 1))
    #define va end(ap)
                          (ap = (char*)0)
    extern int TARGET_VSNPRINTF(char*, size_t, const char*, char*);
#else
    #ifndef _lint
       #include <stdarg.h> // skip this GNU header for win32 #include <stdio.h>
    #endif
    #define TARGET_VSNPRINTF(a,b,c,d) vsprintf(a,c,d)
#endif
// Like stdio.h printf, but writes to UTL_FileSource
// *WARNING* Don't write too much data -- 500 chars max!
int UTL_FileUser::printf (const char * fmt, ...) //lint !e1916
#ifdef _lint
```





```
// to make lint happy
        assert(1);
    return 0; // the real function is just to crazy for lint
#else
#ifdef WIN32 // definition from Visual C++ stdio.h, with defines resolved (different from GNU)
    #define target_va_list char*
#else
    #define target va_list va_list
#endif
    char tmp[500];
    target_va_list ap;
    va_start(ap, fmt);
    int ret = TARGET_VSNPRINTF(tmp, sizeof(tmp), fmt, ap);
       assert( ret >= 0 && ret <= (int)(sizeof(tmp)) );
    va_end (ap);
    puts((const char *)tmp);
    return (ret);
#endif
// write a null-terminated string, with buffering
void UTL FileUser::puts( const char * txt )
    const char * p = txt;
    while( *p )
    {
        if( bufferedChars >= theLength )
            flush();
        theBuffer[bufferedChars++] = (unsigned char)*p++;
    }
}
// write data
void UTL_FileUser::write(unsigned char *buffer, UR_UINT16 length)
    flush();
    sendFrame(buffer,length);
// Ensure that all data has been sent
void UTL_FileUser::flush(void)
    if( bufferedChars )
    {
        sendFrame(theBuffer, bufferedChars);
        bufferedChars = 0;
}
UTL FileUser::UTL_FileUser(unsigned char *buffer, UR_UINT16 length)
        theBuffer = buffer;
        theLength = length;
    bufferedChars = 0;
}
UTL FileUser::~UTL_FileUser()
```